```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
```

```
  CCCCCCCC    OOOOOO    NN      NN    IIIIII    NN      NN  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR    RRRRRRRR
  CCCCCCCC    OOOOOO    NN      NN    IIIIII    NN      NN  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR    RRRRRRRR
CC          OO      OO  NN      NN      II      NN      NN      TT      EE          RR      RR  RR      RR
CC          OO      OO  NN      NN      II      NN      NN      TT      EE          RR      RR  RR      RR
CC          OO      OO  NNNN    NN      II      NNNN    NN      TT      EE          RR      RR  RR      RR
CC          OO      OO  NNNN    NN      II      NNNN    NN      TT      EE          RR      RR  RR      RR
CC          OO      OO  NN  NN  NN      II      NN  NN  NN      TT      EEEEEEEE    RRRRRRRR    RRRRRRRR
CC          OO      OO  NN  NN  NN      II      NN  NN  NN      TT      EEEEEEEE    RRRRRRRR    RRRRRRRR
CC          OO      OO  NN    NNNN      II      NN    NNNN      TT      EE          RR  RR      RR  RR
CC          OO      OO  NN    NNNN      II      NN    NNNN      TT      EE          RR  RR      RR  RR
CC          OO      OO  NN      NN      II      NN      NN      TT      EE          RR    RR    RR    RR      ....
CC          OO      OO  NN      NN      II      NN      NN      TT      EE          RR    RR    RR    RR      ....
  CCCCCCCC    OOOOOO    NN      NN    IIIIII    NN      NN      TT      EEEEEEEEEE  RR      RR  RR      RR    ....
  CCCCCCCC    OOOOOO    NN      NN    IIIIII    NN      NN      TT      EEEEEEEEEE  RR      RR  RR      RR    ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  CONINTERR - Connect to interrupt driver
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ;++
0000    29 ;
0000    30 ; FACILITY:
0000    31 ;
0000    32 ;       VAX/VMS Connect to Interrupt Driver
0000    33 ;
0000    34 ; ABSTRACT:
0000    35 ;
0000    36 ;       This driver has the following pieces:
0000    37 ;
0000    38 ;               An FDT routine to process the IO$_CONINTREAD and
0000    39 ;                       IO$_CONINTWRITE functions
0000    40 ;               A skeletal start device routine
0000    41 ;               A skeletal device initialization routine
0000    42 ;               A skeletal interrupt service routine
0000    43 ;               A skeletal cancel I/O routine
0000    44 ;
0000    45 ; AUTHOR:
0000    46 ;
0000    47 ;       Carol Peters    20-Aug-1979
0000    48 ;
0000    49 ; REVISION HISTORY:
0000    50 ;       V03-006 ROW0406         Ralph O. Weber          24-JUL-1984
0000    51 ;               Cause DEV$V_AVL in UCB$L_DEVCHAR to be set for devices
0000    52 ;               controlled by this driver.
0000    53 ;
0000    54 ;       V03-005 ROW64023        Ralph O. Weber          14-FEB-1984
0000    55 ;               Fix ERROR_DEALSPTS so that it tests CIN$L_SPTCOUNT for zero
0000    56 ;               before calling EXE$DEAL_SPTS.
0000    57 ;
```

```
0000     58 ;       V03-004 ROW56322        Ralph O. Weber          10-JUN-1983
0000     59 ;       ECO 3   Make several corrections (most of which correct mistakes made
0000     60 ;               in V3.2 -- ECO02):
0000     61 ;                  o    Fix buffer double mapping code (20$ after DOUBLE_MAP)
0000     62 ;                       to map kernel-read access if IO$_CONINIREAD function
0000     63 ;                       code is used.
0000     64 ;                  o    Change AST count test mask to a literal not an address.
0000     65 ;                  o    Change UCB reference in CI_CANCEL from R0 to R5.
0000     66 ;
0000     67 ;       V03-003 ROW0126         Ralph O. Weber          19-SEP-1982
0000     68 ;               Delete CI_DUMMY_RET.  It's never used and its presence
0000     69 ;               confuses the casual observer regarding how the driver works.
0000     70 ;
0000     71 ;       V03-002 ROW41224        Ralph O. Weber           7-JUL-1982
0000     72 ;       ECO 2   Force UCB$M_CI_USECAL, UCB$M_CI_INIDEV, UCB$M_CI_START,
0000     73 ;               UCB$M_CI_ISR, and UCB$M_CI_CANCEL to be clear whenever the
0000     74 ;               user buffer and its attendent routines are not present.  Fix
0000     75 ;               CI_INIT_DEVICE to test both UCB$M_CI_USECAL and
0000     76 ;               UCB$M_CI_INIDEV before using the CALLG interface.  Fix
0000     77 ;               CI_CANCEL to test both UCB$M_CI_USECAL and UCB$M_CI_CANCEL
0000     78 ;               before using the CALLG interface.  Correct code near
0000     79 ;               SETUP_ASTS so that UCB$M_CI_REPEAT gets automatically set if
0000     80 ;               and only if P6 (AST_COUNT) is present.  Also streamline this
0000     81 ;               code by removing improperly coded test for negative blocks to
0000     82 ;               preallocate count.  Add sanity check ASSUME statements to
0000     83 ;               guarantee that UCB$M_CI_xxxxxx symbols match CIN$M_xxxxxx,
0000     84 ;               since they are used interchangabely.
0000     85 ;               This correction will be distributed with Version 3.2.
0000     86 ;
0000     87 ;       V03-001 KDM46539        Kathleen D. Morse        4-Jun-1982
0000     88 ;               Set the size into the pool block allocated via
0000     89 ;               EXE$ALONONPAGED.  Also, fix some comments.
0000     90 ;
0000     91 ;--
```

```
                                        .SBTTL  External and local symbol definitions

                                ;
                                ; External symbols
                                ;

                                        $ACBDEF                         ; AST control block
                                        $CINDEF                         ; Connect to interrupt offsets
                                        $CRBDEF                         ; Channel request block
                                        $DCDEF                          ; Device classes and types
                                        $DDBDEF                         ; Device data block
                                        $DEVDEF                         ; Device characteristics bits
                                        $DPTDEF                         ; Device prologue table fields
                                        $DYNDEF                         ; Control block types
                                        $IDBDEF                         ; Interrupt data block
                                        $IODEF                          ; I/O function codes
                                        $IPLDEF                         ; Hardware IPL definitions
                                        $IRPDEF                         ; I/O request packet
                                        $PCBDEF                         ; Process control block fields
                                        $PRDEF                          ; Processor registers
                                        $PRIDEF                         ; Process priorities
                                        $PRVDEF                         ; User privilege codes
                                        $PSLDEF                         ; Program status longword
                                        $PTEDEF                         ; Page table entry definitions
                                        $RBMDEF                         ; Realtime bit map block
                                        $SSDEF                          ; System status codes
                                        $UCBDEF                         ; Unit control block
                                        $VADEF                          ; Virtual address fields
                                        $VECDEF                         ; Interrupt vector block

                                ;
                                ; Local symbols
                                ;

                                ;
                                ; Argument list (AP) offsets for device-dependent QIO parameters
                                ;

00000000                        P1              = 0                     ; First QIO parameter
00000004                        P2              = 4                     ; Second QIO parameter
00000008                        P3              = 8                     ; Third QIO parameter
0000000C                        P4              = 12                    ; Fourth QIO parameter
00000010                        P5              = 16                    ; Fifth QIO parameter
00000014                        P6              = 20                    ; Sixth QIO parameter

00000000                        BUFFER_DESC     = P1                    ; Address of descriptor for the
                                                                        ; connect to interrupt buffer.
00000004                        ENTRY_LIST      = P2                    ; List of entry points.
00000008                        FLAGS           = P3                    ; Connect to interrupt flags.
0000000C                        AST_ROUTINE     = P4                    ; Address of associated AST
                                                                        ; routine.
00000010                        AST_PARAMETER   = P5                    ; Address of AST parameter.
00000014                        AST_COUNT       = P6                    ; Number of AST control blocks
                                                                        ; to preallocate.

                                ;
                                ; Added UCB fields for connect to interrupt functions.
```

```
                     0000    150 ;
                     0000    151
                     0000    152         $DEFINI UCB
                     0000    153
00000044             0000    154         .       = UCB$L_DEVDEPEND       ; Set to device dependent field.
                     0044    155
                     0044    156 $VIELD  UCB,0,<-                        ; Define characteristics:
                     0044    157                 <CI_EFN,,M>,-           ; Set event flag on interrupt.
                     0044    158                 <CI_USECAL,,M>,-        ; Use CALL interface.
                     0044    159                 <CI_REPEAT,,M>,-        ; Repeat delivery of interrupts.
                     0044    160                 <CI_AST,,M>,-           ; Queue AST on interrupt.
                     0044    161                 <CI_INIDEV,,M>,-        ; Device init routine present.
                     0044    162                 <CI_START,,M>,-         ; Start device routine present.
                     0044    163                 <CI_ISR,,M>,-           ; ISR routine present.
                     0044    164                 <CI_CANCEL,,M>,-        ; Cancel I/O routine present.
                     0044    165                 <CI_UCBFRK,,M>,-        ; Fork on UCB has occurred.
                     0044    166         >
                     0044    167
                     0044    168         ASSUME  UCB$M_CI_EFN    EQ CIN$M_EFN
                     0044    169         ASSUME  UCB$M_CI_USECAL EQ CIN$M_USECAL
                     0044    170         ASSUME  UCB$M_CI_REPEAT EQ CIN$M_REPEAT
                     0044    171         ASSUME  UCB$M_CI_AST    EQ CIN$M_AST
                     0044    172         ASSUME  UCB$M_CI_INIDEV EQ CIN$M_INIDEV
                     0044    173         ASSUME  UCB$M_CI_START  EQ CIN$M_START
                     0044    174         ASSUME  UCB$M_CI_ISR    EQ CIN$M_ISR
                     0044    175         ASSUME  UCB$M_CI_CANCEL EQ CIN$M_CANCEL
                     0044    176
00000090             0044    177         .       = UCB$K_LENGTH          ; Set offset to end of standard
                     0090    178                                         ; UCB.
                     0090    179
                     0090    180 $DEF    UCB$Q_CI_BUFDSC                 ; Buffer descriptor parameter.
00000094             0090    181                 .BLKL   1
00000098             0094    182                 .BLKL   1
                     0098    183 $DEF    UCB$B_CI_ASTMOD                 ; Mode at which to deliver AST.
00000099             0098    184                 .BLKB   1
                     0099    185 $DEF    UCB$B_CI_SPARE                  ; Spare byte.
0000009A             0099    186                 .BLKB   1
                     009A    187 $DEF    UCB$W_CI_EFNUM                  ; Event flag number.
0000009C             009A    188                 .BLKW   1
                     009C    189 $DEF    UCB$L_CI_AST                    ; Address of AST routine.
000000A0             009C    190                 .BLKL   1
                     00A0    191 $DEF    UCB$L_CI_ASTPRM                 ; AST parameter.
000000A4             00A0    192                 .BLKL   1
                     00A4    193 $DEF    UCB$W_CI_ACBCNT                 ; Number of AST blocks to
000000A6             00A4    194                 .BLKW   1                  ; preallocate.
                     00A6    195 $DEF    UCB$W_CI_ACBNOW                 ; Count of AST blocks currently
000000A8             00A6    196                 .BLKW   1                  ; allocated.
                     00A8    197 $DEF    UCB$L_CI_AFLINK                 ; Forward link to ACB list.
000000AC             00A8    198                 .BLKL   1
                     00AC    199 $DEF    UCB$L_CI_ABLINK                 ; Backward link to ACB list.
000000B0             00AC    200                 .BLKL   1
                     00B0    201 $DEF    UCB$L_CI_PCB                    ; Address of process' PCB.
000000B4             00B0    202                 .BLKL   1
                     00B4    203 $DEF    UCB$Q_CI_SPTDSC                 ; System page table descriptor
000000B8             00B4    204                 .BLKL   1                  ; for user buffer mapping.
000000BC             00B8    205                 .BLKL   1                  ; Stores SPT count and VPN
                     00BC    206                                         ; of starting page of buffer.
```

```
                00BC    207 $DEF    UCB$L_CI_INIDEV                 ; Address of user-specified
000000C0        00BC    208         .BLKL   1                       ; device initialization routine.
                00C0    209 $DEF    UCB$L_CI_START                  ; Address of user-specified
000000C4        00C0    210         .BLKL   1                       ; start device routine.
                00C4    211 $DEF    UCB$L_CI_STACAL                 ; Address of user-specified
000000C8        00C4    212         .BLKL   1                       ; start device routine using
                00C8    213                                         ; CALL interface.
                00C8    214 $DEF    UCB$L_CI_ISR                    ; Address of user-specified
000000CC        00C8    215         .BLKL   1                       ; interrupt service routine.
                00CC    216 $DEF    UCB$L_CI_ISRCAL                 ; Address of user-specified
000000D0        00CC    217         .BLKL   1                       ; interrupt service routine
                00D0    218                                         ; using CALL interface.
                00D0    219 $DEF    UCB$L_CI_CANCEL                 ; Address of user-specified
000000D4        00D0    220         .BLKL   1                       ; cancel I/O routine.
                00D4    221
                00D4    222 ;
                00D4    223 ; The next set of fields must be in exactly the order you see them.
                00D4    224 ;
                00D4    225
                00D4    226 $EQU    UCB$K_CI_STARGC 4               ; Number of arguments for
                00D4    227                                         ; start device routine.
                00D4    228 $DEF    UCB$L_CI_STARGC                 ; Argument count for start
000000D8        00D4    229         .BLKL   1                       ; device routine.
                00D8    230 $DEF    UCB$L_CI_STARG1                 ; First start device argument.
000000DC        00D8    231         .BLKL   1
                00DC    232 $DEF    UCB$L_CI_STARG2                 ; Second start device argument.
000000E0        00DC    233         .BLKL   1
                00E0    234 $DEF    UCB$L_CI_STARG3                 ; Third start device argument.
000000E4        00E0    235         .BLKL   1
                00E4    236 $DEF    UCB$L_CI_STARG4                 ; Fourth start device argument.
000000E8        00E4    237         .BLKL   1
                00E8    238
                00E8    239 ;
                00E8    240 ; The next set of fields must be in exactly the order you see them.
                00E8    241 ;
                00E8    242
                00E8    243 $EQU    UCB$K_CI_ISARGC 5               ; Number of arguments for
                00E8    244                                         ; interrupt service routine.
                00E8    245 $DEF    UCB$L_CI_ISARGC                 ; Argument count for ISR.
000000EC        00E8    246         .BLKL   1
                00EC    247 $DEF    UCB$L_CI_ISARG1                 ; First argument for ISR.
000000F0        00EC    248         .BLKL   1
                00F0    249 $DEF    UCB$L_CI_ISARG2                 ; Second argument for ISR.
000000F4        00F0    250         .BLKL   1
                00F4    251 $DEF    UCB$L_CI_ISARG3                 ; Third argument for ISR.
000000F8        00F4    252         .BLKL   1
                00F8    253 $DEF    UCB$L_CI_ISARG4                 ; Fourth argument for ISR.
000000FC        00F8    254         .BLKL   1
                00FC    255 $DEF    UCB$L_CI_ISARG5                 ; Fifth argument for ISR.
00000100        00FC    256         .BLKL   1
                0100    257
                0100    258 $DEF    UCB$K_CI_LENGTH                 ; Length of CI UCB.
                0100    259
                0100    260         $DEFEND UCB
                0000    261
                0000    262 ;
                0000    263 ; Other constants
```

```
0000   264 ;
```

```
                        .SBTTL  Standard tables
                ;
                ; Driver prologue table
                ;

                DPTAB   -                                       ; DPT-creation macro
                        END=CI_END,-                            ; End of driver label
                        ADAPTER=UBA,-                           ; Adapter type
                        UCBSIZE=<UCB$K_CI_LENGTH>,-             ; Length of UCB
                        NAME=CONINTERR                  ; Driver name
                DPT_STORE INIT                                  ; Start of load
                                                                ; initialization table
                DPT_STORE UCB,UCB$B_FIPL,B,6                    ; Driver fork IPL
                DPT_STORE UCB,UCB$B_DIPL,B,22                   ; Device interrupt IPL
                DPT_STORE UCB,UCB$L_DEVDEPEND,L,0               ; Clear device dependent
                                                                ; bits.
                DPT_STORE UCB,UCB$L_DEVCHAR,L,<-; Set device characteristics:
                                DEV$M_AVL -     ;       available for use
                              ! DEV$M_RTM -     ;       real-time device
                              >
                DPT_STORE REINIT                                ; Start of reload
                                                                ; initialization table
                DPT_STORE DDB,DDB$L_DDT,D,CI$DDT                ; Address of DDT
                DPT_STORE CRB,CRB$L_INTD+4,D,-                  ; Address of interrupt
                        CI_INTERRUPT                            ; service routine
                DPT_STORE CRB,-                                 ; Address of controller
                        CRB$L_INTD+VEC$L_INITIAL,-              ; initialization routine
                        D,CI_INIT_DEVICE
                DPT_STORE UCB,UCB$L_CI_INIDEV,D,-               ; Address of user's
                        CI_DUMMY_RSB                            ; device initialization
                                                                ; routine.
                DPT_STORE UCB,UCB$L_CI_START,D,-                ; Address of user's
                        CI_DUMMY_RSB                            ; start I/O routine.
                DPT_STORE UCB,UCB$L_CI_ISR,D,-                  ; Address of user's
                        CI_DUMMY_RSB                            ; interrupt service
                                                                ; routine.
                DPT_STORE UCB,UCB$L_CI_CANCEL,D,-               ; Address of user's
                        CI_DUMMY_RSB                            ; cancel I/O routine.

                DPT_STORE END                                   ; End of initialization
                                                                ; tables.

                ;
                ; Driver dispatch table
                ;

                DDTAB   -                                       ; DDT-creation macro
                        DEVNAM=CI,-                             ; Name of device
                        START=CI_START,-                        ; Start I/O routine
                        FUNCTB=CI_FUNCTABLE,-                   ; FDT address
                        CANCEL=CI_CANCEL                        ; Cancel I/O routine

                ;
                ; Function dispatch table
                ;
```

```
                     ;
                     ; FDT for driver
CI_FUNCTABLE:
        FUNCTAB ,-                                   ; Valid I/O functions
                <CONINTREAD,-                        ; Connect to interrrupt
                 CONINTWRITE>                        ; read and write codes.
        FUNCTAB
        FUNCTAB CI_CONNECT,-                         ; FDT connect to
                <CONINTREAD,-                        ; interrupt readonly
                 CONINTWRITE>                        ; and write.
                     ;
        .SHOW   EXPANSIONS
```

```
0054    335          .SBTTL  CI_INIT_DEVICE, Controller initialization routine
0054    336
0054    337 ;++
0054    338 ; CI_INIT_DEVICE, Readies controller for I/O operations
0054    339 ;
0054    340 ; Functional description:
0054    341 ;
0054    342 ;       The operating system calls this routine in 3 places:
0054    343 ;
0054    344 ;               at system startup
0054    345 ;               during driver loading and reloading
0054    346 ;               during recovery from a power failure
0054    347 ;
0054    348 ;       This routine sets the device online, and marks the device
0054    349 ;       as the owner of the controller. Then the routine calls a
0054    350 ;       user-specified device initialization routine. The FDT routine
0054    351 ;       CI_CONNECT loads a user-specified routine address into the
0054    352 ;       relevant UCB field.
0054    353 ;
0054    354 ;       The selection of the CALLS or JSB path is via a bit setting
0054    355 ;       in the UCB. When the user's routine is called, R0 contains the
0054    356 ;       address of the UCB; registers R4-R8 are unchanged; for a CALL
0054    357 ;       interface, the argument list is as follows:
0054    358 ;
0054    359 ;               0(AP)  - argument count of #5.
0054    360 ;               4(AP)  - the address of the CSR
0054    361 ;               8(AP)  - the address of the IDB
0054    362 ;              12(AP)  - the address of the DDB
0054    363 ;              16(AP)  - the address of the CRB
0054    364 ;              20(AP)  - the address of the UCB
0054    365 ;
0054    366 ; Inputs:
0054    367 ;
0054    368 ;       R4      - address of the CSR (controller status register)
0054    369 ;       R5      - address of the IDB (interrupt data block)
0054    370 ;       R6      - address of the DDB (device data block)
0054    371 ;       R8      - address of the CRB (channel request block)
0054    372 ;
0054    373 ; Implicit inputs:
0054    374 ;
0054    375 ;       UCB$V_CI_USECAL bit is set in UCB$L_DEVDEPEND if the CALLS
0054    376 ;       interface is desired.
0054    377 ;
0054    378 ;       UCB$L_CI_INIDEV contains the address of the user-specified
0054    379 ;       device initialization routine.
0054    380 ;
0054    381 ; Outputs:
0054    382 ;
0054    383 ;       The routine must preserve all registers except R0-R3.
0054    384 ;
0054    385 ;--
0054    386
0054    387 CI_INIT_DEVICE:                            ; Initialize controller
0054    388
0054    389 ;
0054    390 ; Mark the device as online in the UCB, and indicate in the IDB that
0054    391 ; the device is the owner of the controller.
```

```
                     0054   392 ;
                     0054   393
 50   04 A6   D0     0054   394          MOVL    DDB$L_UCB(R6),R0          ; Get address of UCB.
       10   A8       0058   395          BISW    #UCB$M_ONLINE,-           ; Mark device online.
          64 A0       005A   396                  UCB$W_STS(R0)
 04 A5    50   D0     005C   397          MOVL    R0,IDB$L_OWNER(R5)        ; Set device as controller
                     0060   398                                            ; owner.
                     0060   399
                     0060   400 ;
                     0060   401 ; Now call the user-specified device initialization routine.
                     0060   402 ;
                     0060   403
       01   E1       0060   404          BBC     #UCB$V_CI_USECAL,-        ; Branch to JSB code if user
     15 44 A0       0062   405                  UCB$L_DEVDEPEND(R0),10$   ; didn't request CALL interface.
10 44 A0    04   E1  0065   406          BBC     #UCB$V_CI_INIDEV, -       ; Branch to JSB code if user
                     006A   407                  UCB$L_DEVDEPEND(R0), 10$; initization routine doesn't exist.
                     006A   408
                     006A   409 ;
                     006A   410 ; Load the input registers onto the argument stack and CALLS the
                     006A   411 ; user-specified initialization routine.
                     006A   412 ;
                     006A   413
          50   DD     006A   414          PUSHL   R0                        ; Push address of UCB.
          58   DD     006C   415          PUSHL   R8                        ; Push address of CRB.
          56   DD     006E   416          PUSHL   R6                        ; Push address of DDB.
          55   DD     0070   417          PUSHL   R5                        ; Push address of IDB.
          54   DD     0072   418          PUSHL   R4                        ; Push address of CSR.
00BC DO   05   FB     0074   419          CALLS   #5,@UCB$L_CI_INIDEV(R0)   ; Call user-specified device
                     0079   420                                            ; initialization routine.
          05   05     0079   421          RSB                               ; Return.
                     007A   422
                     007A   423 ;
                     007A   424 ; Just JSB to the user-specified initialization routine.
                     007A   425 ;
                     007A   426
                     007A   427 10$:                                        ; JSB path.
00BC DO   16   DO     007A   428          JSB     @UCB$L_CI_INIDEV(R0)      ; JSB to user-specified device
                     007E   429                                            ; initialization routine.
          05   05     007E   430          RSB                               ; Return.
```

```
                        .SBTTL  CI_CONNECT, Connect the process to an interrupt

                ;++
                ; CI_CONNECT, FDT routine that establishes an interrupt handler
                ;
                ; Functional description:
                ;
                ;       This routine gains control at IPL IPL$_ASTDEL.
                ;
                ;       This routine puts the process in control of the device in the
                ;       following steps:
                ;
                ;       1. Validate and clear the event flag.
                ;       2. If the buffer descriptor describes a non-zero length buffer,
                ;          check access to entry point list, confirm that process has
                ;          CMKRNL privilege, and lock buffer pages in memory.
                ;       3. Double map the buffer to system space.
                ;       4. Setup for CIDRIVER calling of process-supplied kernel mode
                ;          routines for device control.
                ;       5. If the address of an AST routine is supplied, allocate
                ;          and initialize a specified number of AST control blocks.
                ;       6. Queue the IRP to the driver; this activates the driver's
                ;          start I/O routine, which passes control to the process-start
                ;          I/O routine (if any).
                ;
                ; Inputs:
                ;
                ;       R0-R2   - scratch registers
                ;       R3      - address of the IRP (I/O request packet)
                ;       R4      - address of the PCB (process control block)
                ;       R5      - address of the UCB (unit control block)
                ;       R6      - address of the CCB (channel control block)
                ;       R7      - bit number of the I/O function code
                ;       R8      - address of the FDT table entry for this routine
                ;       R9-R11  - scratch registers
                ;       AP      - address of the 1st function dependent QIO parameter
                ;
                ;       6 parameters can be specified; they are as follows:
                ;
                ;               BUFFER_DESC(AP)         - buffer descriptor
                ;               ENTRY_LIST(AP)          - address of entry point list
                ;               FLAGS(AP)               - flags
                ;                       low word is pure flags
                ;                       high word is event flag number
                ;               AST_ROUTINE(AP)         - AST address
                ;               AST_PARAMETER(AP)       - AST parameter
                ;               AST_COUNT(AP)           - count of AST control blocks
                ;                                         to preallocate
                ;
                ;       The ENTRY_LIST parameter is the address of a 4-longword block
                ;       that contains offsets into the user buffer:
                ;
                ;               CIN$L_INIDEV    - offset to device init routine
                ;               CIN$L_START     - offset to start device routine
                ;               CIN$L_ISR       - offset to interrupt service routine
                ;               CIN$L_CANCEL    - offset to cancel I/O routine
                ;
```

```
                     007F    489 ;       The FLAGS parameter has the following flags settings:
                     007F    490 ;
                     007F    491 ;               CIN$M_EFN       - set event flag on interrupt
                     007F    492 ;               CIN$M_USECAL    - use a CALLS interface to user routines
                     007F    493 ;               CIN$M_REPEAT    - repeatedly report interrupts
                     007F    494 ;               CIN$M_AST       - queue AST on interrupt
                     007F    495 ;               CIN$M_INIDEV    - initialize device routine in buffer
                     007F    496 ;               CIN$M_START     - start device routine in buffer
                     007F    497 ;               CIN$M_ISR       - interrupt service routine in buffer
                     007F    498 ;               CIN$M_CANCEL    - cancel I/O routine in buffer
                     007F    499 ;
                     007F    500 ;               CIN$V_EFNUM     - offset to event flag number
                     007F    501 ;               CIN$S_EFNUM     - size of event flag number field
                     007F    502 ;
                     007F    503 ; Outputs:
                     007F    504 ;
                     007F    505 ;       The routine must preserve all registers except R0-R2, and
                     007F    506 ;       R9-R11.
                     007F    507 ;
                     007F    508 ;--
                     007F    509
                     007F    510 CI_CONNECT:                                 ; Establish a handler.
50   204C 8F   3C    007F    511         MOVZWL  #SS$_DISCONNECT,R0          ; Assume connect in progress
          08   E0    0084    512         BBS     #UCB$V_BSY,-                ; Branch if connect
     34 64 A5         0086    513                 UCB$W_STS(R5),10$           ; is in progress.
00B0 C5    54   D0    0089    514         MOVL    R4,UCB$L_CI_PCB(R5)         ; Save the process PCB.
       08 AC   B0    008E    515         MOVW    FLAGS(AP),-                 ; Store flags bits in the UCB.
       44 A5           0091    516                 UCB$L_DEVDEPEND(R5)
                       0093    517
                       0093    518 ;
                       0093    519 ; Force the AST wanted flag to agree with whether an AST address
                       0093    520 ; was specified by the caller.
                       0093    521 ;
                       0093    522
          08   AA      0093    523         BICW    #UCB$M_CI_AST,-             ; Assume AST's not wanted.
       44 A5           0095    524                 UCB$L_DEVDEPEND(R5)         ;
       0C AC   D5      0097    525         TSTL    AST_ROUTINE(AP)             ; AST addr specified?
          04   13      009A    526         BEQL    5$                          ; Branch if not.
          08   A8      009C    527         BISW    #UCB$M_CI_AST,-             ; Else force AST bit set.
       44 A5           009E    528                 UCB$L_DEVDEPEND(R5)         ;
                       00A0    529
                       00A0    530 5$:
                       00A0    531
                       00A0    532 ;
                       00A0    533 ; If the user specified an event flag to be posted in the event of an
                       00A0    534 ; interrupt, clear the event flag, thereby checking for an invalid
                       00A0    535 ; event flag specification.
                       00A0    536 ;
                       00A0    537
08 AC      00   E1      00A0    538         BBC     #CIN$V_EFN,FLAGS(AP),-      ; Don't check event flag unless
             1B         00A4    539                 20$                         ; requested.
             08   BB    00A5    540         PUSHR   #^M<R3>                     ; Save the IRP address.
             10   EF    00A7    541         EXTZV   #CIN$V_EFNUM,-              ; Extract the event flag
             10         00A9    542                 #CIN$S_EFNUM,-              ; number from the high flags
   53    08 AC           00AA    543                 FLAGS(AP),R3                ; word.
009A C5    53   B0      00AD    544         MOVW    R3,UCB$W_CI_EFNUM(R5)       ; Store event flag number in
                       00B2    545                                             ; the UCB.
```

```
          00000000'GF   16  00B2        JSB     G^SCH$CLREF              ; Clear and test event flag.
                   08   BA  00B8        POPR    #^M<R3>                  ; Restore IRP address.
                03 50   E8  00BA        BLBS    R0,20$                   ; Branch forward on success.
                 01F5   31  00BD  10$:  BRW     ERROR                    ; Stop with error.
                            00C0
                            00C0  ;
                            00C0  ; See if the user specified a buffer. If yes, and the buffer is of
                            00C0  ; a finite length, go on to look at the entry point list. Otherwise,
                            00C0  ; just proceed to AST setup code.
                            00C0  ;
                            00C0
                            00C0  20$:
          00B4 C5      7C   00C0        CLRQ    UCB$Q_CI_SPTDSC(R5)      ; Clear buffer descriptor in
                            00C4                                         ; UCB.
          5A     6C    D0   00C4        MOVL    BUFFER_DESC(AP),R10      ; Get buffer descriptor.
                 04    13   00C7        BEQL    30$                      ; Branch if no descriptor.
                 6A    B5   00C9        TSTW    (R10)                    ; Is buffer non zero length?
                 0B    12   00CB        BNEQ    40$                      ; Yes. Go check entry list.
                            00CD
                            00CD  30$:
 44 A5   000000F2 8F   CA   00CD        BICL    #<UCB$M_CI_USECAL ! -    ; Can't use the CALL interface to
                            00D5                UCB$M_CI_INIDEV ! -      ; routines which are not there.
                            00D5                UCB$M_CI_START ! -
                            00D5                UCB$M_CI_ISR ! -
                            00D5                UCB$M_CI_CANCEL>, -
                            00D5                UCB$L_DEVDEPEND(R5)
               0152    31   00D5        BRW     SETUP_ASTS               ; Skip access checks if length
                            00D8
                            00D8  ;
                            00D8  ; Return error if buffer size exceeds 65767 bytes.
                            00D8  ;
                            00D8
                            00D8  40$:
          50     14    3C   00D8        MOVZWL  #SS$_BADPARAM,R0         ; Assume error.
   0000FFFF 8F   6A    D1   00DB        CMPL    (R10),#^XFFFF            ; Byte count .ge. 65767?
                 D9    14   00E2        BGTR    10$                      ; Branch if so.
                            00E4
                            00E4  ;
                            00E4  ; Validate read access to the entry point list.
                            00E4  ;
                            00E4
          50     0C    3C   00E4        MOVZWL  #SS$_ACCVIO,R0           ; Assume read access failure.
       5B    04 AC     D0   00E7        MOVL    ENTRY_LIST(AP),R11       ; Get address of entry list.
                            00EB        IFRD    #4*4,(R11),50$           ; Branch forward if process has
    6B   10    00      0C   00EB                PROBER  #0,#4*4,(R11)
                 03    12   00EF                BNEQ    50$
                            00F1
                            00F1                                         ; read access to list.
               01C1    31   00F1        BRW     ERROR                    ; Otherwise, stop with error.
                            00F4
                            00F4  ;
                            00F4  ; Check for change mode to kernel privilege, without which, executing
                            00F4  ; a routine in kernel mode (either as an ISR, device initialization,
                            00F4  ; etc.) is not permitted.
                            00F4  ;
                            00F4
                            00F4  50$:
```

```
                                                I 15
                     50   24   3C  00F4   600          MOVZWL  #SS$_NOPRIV,R0          ; Assume privilege violation.
                                  00F7   601          IFPRIV  CMKRNL,LOCK_PAGES       ; If process is sufficiently
                                  00F7                .IF DIF <CMKRNL>,<R1>
                                  00F7                .IF DIF <CMKRNL>,<R2>
           03 0084 C4    00   E0  00F7                BBS     #PRV$V_CMKRNL,PCB$Q_PRIV(R4),LOCK_PAGES
                                  00FD                .IFF
                                  00FD                BBS     CMKRNL,PCB$Q_PRIV(R4),LOCK_PAGES
                                  00FD                .ENDC
                                  00FD                .IFF
                                  00FD                BBS     CMKRNL,PCB$Q_PRIV(R4),LOCK_PAGES
                                  00FD                .ENDC
                                  00FD
                                  00FD   602                                          ; privileged, proceed.
                         01B5     31   00FD   603          BRW     ERROR              ; Otherwise, stop now.
                                       0100   604
                                       0100   605 ;
                                       0100   606 ; Lock down the user pages so they can't be paged out during interrupt
                                       0100   607 ; servicing.
                                       0100   608 ;
                                       0100   609 ; The register setup before calling VMS to lock the pages is as follows:
                                       0100   610 ;
                                       0100   611 ;       R0      - buffer address
                                       0100   612 ;       R1      - buffer length in bytes
                                       0100   613 ;       R3      - address of the IRP
                                       0100   614 ;       R4      - address of the PCB
                                       0100   615 ;       R6      - address of the CCB
                                       0100   616 ;       R11     - entry list address
                                       0100   617 ;
                                       0100   618 ; The locking routines return the address of the page table entry for
                                       0100   619 ; the first page in the user's buffer in R1 and in IRP$L_SVAPTE.
                                       0100   620 ;
                                       0100   621
                                       0100   622 LOCK_PAGES:
                51    6A     3C      0100   623          MOVZWL  (R10),R1                ; Get buffer length.
              50 04   AA     D0      0103   624          MOVL    4(R10),R0               ; Get buffer address.
                      00     EF      0107   625          EXTZV   #IRP$V_FCODE,-          ; Get the function code.
                      06             0109   626                  #IRP$S_FCODE,-
              59 20   A3             010A   627                  IRP$W_FUNC(R3),R9
                 59   3D     D1      010D   628          CMPL    #IO$_CONINTWRITE,R9     ; Is it a write?
                      08     13      0110   629          BEQL    10$                     ; Yes, branch to write lock.
00000000'GF           16     0112   630          JSB     G^EXE$WRITELOCK         ; Otherwise, check for read
                                       0118   631                                  ; access and lock pages.
                      06     11      0118   632          BRB     DOUBLE_MAP              ; The routine only returns if
                                       011A   633                                  ; successful; branch forward.
                                       011A   634
                                       011A   635 10$:
00000000'GF           16     011A   636          JSB     G^EXE$MODIFYLOCK        ; Check for modify access and
                                       0120   637                                  ; lock pages. Only return is
                                       0120   638                                  ; success. Failure aborts or
                                       0120   639                                  ; backs out I/O request to wait
                                       0120   640                                  ; for paging activity.
                                       0120   641
                                       0120   642 ;
                                       0120   643 ; Double map the buffer into system page table entries. If SPTs are not
                                       0120   644 ; available, return with error (I/O post will unlock the pages).
                                       0120   645 ;
                                       0120   646
```

```
                                0120   647 DOUBLE_MAP:
          52   00B4 C5   9E   0120   648         MOVAB   UCB$Q_CI_SPTDSC(R5),R2  ; Get address in UCB where
                              0125   649                                        ; the SPT descriptor will go.
           50    6A   3C   0125   650         MOVZWL  (R10),R0                   ; Get # bytes to double map
     51  04 AA    09   00   EF   0128   651         EXTZV   #0,#9,4(R10),R1            ; Get byte offset of buffer
       50   01FF C041   9E   012E   652         MOVAB   ^X1FF(R0)[R1],R0           ; Compute # of bytes to map
      62   50    F7 8F   78   0134   653         ASHL    #-9,R0,-                   ; Convert # bytes to pages
                              0139   654                 CIN$L_SPTCOUNT(R2)         ;
                              0139   655
                              0139   656 10$:
                              0139   657         DSBINT  UCB$B_FIPL(R5)             ; Raise to driver fork IPL.
                              0139                .IF B
                          7E  12   DB 0139                MFPR    S^#PR$_IPL,-(SP)
                              013C                .IFF
                              013C                MFPR    S^#PR$_IPL,
                              013C                .ENDC
                              013C                .IF B   UCB$B_FIPL(R5)
                              013C                MTPR    #31,S^#PR$_IPL
                              013C                .IFF
                       12  0B A5   DA 013C                MTPR    UCB$B_FIPL(R5),S^#PR$_IPL
                              0140                .ENDC
                              0140
          00000486'GF      16   0140   658         JSB     G^EXE$ALLOC_SPTS           ; Allocate the SPTs.
                06 50      E8   0146   659         BLBS    R0,20$                     ; Branch forward on success.
                              0149   660         ENBINT                             ; Drop IPL back down.
                              0149                .IF B
                       12  8E   DA 0149                MTPR    (SP)+,S^#PR$_IPL
                              014C                .IFF
                              014C                MTPR    ,S^#PR$_IPL
                              014C                .ENDC
                              014C
                    0166      31   014C   661         BRW     ERROR                      ; Otherwise, stop with error.
                              014F   662
                              014F   663 ;
                              014F   664 ; R2 now contains a descriptor:
                              014F   665 ;
                              014F   666 ;       CIN$L_SPTCOUNT(R2)      - number of SPTs allocated
                              014F   667 ;       CIN$L_STARTVPN(R2)      - starting virtual page number (VPN)
                              014F   668 ;
                              014F   669 ; Set up the SPTs to address the user buffer. Any errors from now on
                              014F   670 ; must unlock pages and deallocate the SPTs.
                              014F   671 ;
                              014F   672
                              014F   673 20$:
           51   04 AA    D0   014F   674         MOVL    4(R10),R1                  ; Get address of user buffer.
50    10000000 8F    D0   0153   675         MOVL    #<PTE$C_KW>,R0             ; Set write access mask.
         3C   59    D1   015A   676         CMPL    R9,#IO$_CONINTREAD         ; Is this a read?
              07    12   015D   677         BNEQ    30$                        ; No. Branch forward.
50    18000000 8F    D0   015F   678         MOVL    #<PTE$C_KR>,R0             ; Otherwise, restrict to kernel
                              0166   679                                        ; read.
                              0166   680
                              0166   681 30$:
50    80000000 8F    C8   0166   682         BISL    #PTE$M_VALID,R0            ; Set valid bit too.
      000004FC'GF    16   016D   683         JSB     G^EXE$SETUP_SPTS           ; Set up the SPTs.
                              0173   684         ENBINT                             ; Drop IPL back down.
                              0173                .IF B
                       12  8E   DA 0173                MTPR    (SP)+,S^#PR$_IPL
```

```
                     0176                       .IFF
                     0176                       MTPR    ,S^#PR$_IPL
                     0176                       .ENDC
                     0176
                     0176    685
                     0176    686 ;
                     0176    687 ; IPL is now back at 0.
                     0176    688 ;
                     0176    689 ; Get system-mapped address of the user buffer. Registers are:
                     0176    690 ;
                     0176    691 ;       R1      - process address of the user's buffer
                     0176    692 ;       R2      - quadword-descriptor of the SPT count and starting VPN
                     0176    693 ;
                     0176    694
        04 A2   09   78   0176    695         ASHL    #9,CIN$L_STARTVPN(R2),- ; Convert VPN to system
                59           017A    696                 R9                      ; virtual address.
           00   51   F0   017B    697         INSV    R1,#VA$V_BYTE,-         ; Add byte offset into page.
           59   09           017E    698                 #VA$S_BYTE,R9
     59 80000000 8F   C8   0180    699         BISL    #VA$M_SYSTEM,R9         ; Set the system bit.
                     0187    700
                     0187    701 ;
                     0187    702 ; Write proper addresses into driver's
                     0187    703 ;
                     0187    704 ;       device initialization routine
                     0187    705 ;       start device routine
                     0187    706 ;       interrupt service routine
                     0187    707 ;       cancel I/O routine
                     0187    708 ;
                     0187    709 ; Registers used in the following setup are as listed below:
                     0187    710 ;
                     0187    711 ;       R2      - offset to routine in user buffer
                     0187    712 ;       R4      - address of the CRB
                     0187    713 ;       R5      - address of the UCB
                     0187    714 ;       R9      - system-mapped address of the user buffer
                     0187    715 ;       R11     - address of the entry point list
                     0187    716 ;
                     0187    717
                     0187    718 SETUP_ENTRIES:
        54   24 A5   D0   0187    719         MOVL    UCB$L_CRB(R5),R4        ; Get CRB address.
                     018B    720
                     018B    721 ;
                     018B    722 ; Set up for device initialization routine.
                     018B    723 ;
                     018B    724
           04   E1   018B    725         BBC     #CIN$V_INIDEV,-         ; Branch forward if no device
     06 08 AC           018D    726                 FLAGS(AP),10$           ; initialization specified.
00BC C5 59   6B   C1   0190    727         ADDL3   CIN$L_INIDEV(R11),R9,-  ; Set up device initialization
                     0196    728                 UCB$L_CI_INIDEV(R5)     ; routine address.
                     0196    729
                     0196    730 ;
                     0196    731 ; Set up for start I/O routine.
                     0196    732 ;
                     0196    733
                     0196    734 10$:
           05   E1   0196    735         BBC     #CIN$V_START,-          ; Branch forward if no start
     38 08 AC           0198    736                 FLAGS(AP),40$           ; device routine specified.
           01   E1   019B    737         BBC     #CIN$V_USECAL,-         ; Branch forward if not a
```

```
                    12 08 AC          019D   738                  FLAGS(AP),20$              ; CALL interface.
00C4 C5   59    04 AB    C1   01A0   739          ADDL3   CIN$L_START(R11),R9,-   ; Otherwise, store user start
                              01A7   740                  UCB$L_CI_STACAL(R5)        ; device address.
          0000030C'EF    9E   01A7   741          MOVAB   CI_START_CALL,-            ; And store internal label as
              00C0 C5          01AD   742                  UCB$L_CI_START(R5)         ; JSB address.
                   07    11   01B0   743          BRB     30$                        ; Go create argument list.
                              01B2   744
                              01B2   745 20$:                                       ; Normal JSB setup.
00C0 C5   59    04 AB    C1   01B2   746          ADDL3   CIN$L_START(R11),R9,-   ; Set up device start up
                              01B9   747                  UCB$L_CI_START(R5)         ; routine address.
                              01B9   748
                              01B9   749 ;
                              01B9   750 ; Setup canned argument list for the start device routine.
                              01B9   751 ;
                              01B9   752
                              01B9   753 30$:
                        04    D0   01B9   754          MOVL    #UCB$K_CI_STARGC,-         ; Save count of canned
                  00D4 C5          01BB   755                  UCB$L_CI_STARGC(R5)        ; argument list.
              00D8 C5    59    D0   01BE   756          MOVL    R9,UCB$L_CI_STARG1(R5)     ; Start I/O canned list is:
              00DC C5    53    D0   01C3   757          MOVL    R3,UCB$L_CI_STARG2(R5)     ;       buffer address, IRP
                    2C B4    D0   01C8   758          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),-;       address, device CSR
                  00E0 C5          01CB   759                  UCB$L_CI_STARG3(R5)        ;       address, and
              00E4 C5    55    D0   01CE   760          MOVL    R5,UCB$L_CI_STARG4(R5)     ;       the UCB address.
                              01D3   761
                              01D3   762 ;
                              01D3   763 ; Setup for interrupt service routine.
                              01D3   764 ;
                              01D3   765
                              01D3   766 40$:
          08 AC    06    E1   01D3   767          BBC     #CIN$V_ISR,FLAGS(AP),-     ; Branch forward if no ISR
                        40    01D7   768                  70$                        ; was specified.
                        01    E1   01D8   769          BBC     #CIN$V_USECAL,-            ; Branch forward if not a
                    12 08 AC         01DA   770                  FLAGS(AP),50$              ; CALL interface.
00CC C5   59    08 AB    C1   01DD   771          ADDL3   CIN$L_ISR(R11),R9,-     ; Otherwise, store user ISR
                              01E4   772                  UCB$L_CI_ISRCAL(R5)        ; address.
          00000328'EF    9E   01E4   773          MOVAB   CI_ISR_CALL,-              ; And store internal label as
              00C8 C5          01EA   774                  UCB$L_CI_ISR(R5)           ; JSB address.
                   07    11   01ED   775          BRB     60$                        ; Branch to build argument list.
                              01EF   776
                              01EF   777 50$:                                       ; Normal JSB setup.
00C8 C5   59    08 AB    C1   01EF   778          ADDL3   CIN$L_ISR(R11),R9,-     ; Set up interrupt service
                              01F6   779                  UCB$L_CI_ISR(R5)           ; routine address.
                              01F6   780
                              01F6   781 ;
                              01F6   782 ; Setup the canned argument list for the interrupt service routine.
                              01F6   783 ;
                              01F6   784
                              01F6   785 60$:
                        05    D0   01F6   786          MOVL    #UCB$K_CI_ISARGC,-         ; Load count for the canned
                  00E8 C5          01F8   787                  UCB$L_CI_ISARGC(R5)        ; argument list; then load
              00EC C5    59    D0   01FB   788          MOVL    R9,UCB$L_CI_ISARG1(R5)     ;       buffer address,
              00A0 C5    05    DE   0200   789          MOVAL   UCB$L_CI_ASTPRM(R5),-      ;       AST parameter address,
                  00F0 C5          0204   790                  UCB$L_CI_ISARG2(R5)
                              0207   791
                              0207   792          .NOSHOW EXPANSIONS
                              0207   793
                              0207   794          ASSUME  IDB$L_CSR EQ 0
```

```
                2C B4   D0  0207   795          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),-
            00F4 C5         020A   796                  UCB$L_CI_ISARG3(R5)      ;       device CSR address,
                            020D   797
                            020D   798          .SHOW   EXPANSIONS
                            020D   799
                2C A4   D0  020D   800          MOVL    CRB$L_INTD+VEC$L_IDB(R4),-;     the IDB address,
            00F8 C5         0210   801                  UCB$L_CI_ISARG4(R5)      ;       and
        00FC C5   55    D0  0213   802          MOVL    R5,UCB$L_CI_ISARG5(R5)   ;       the UCB address.
                            0218   803
                            0218   804 ;
                            0218   805 ; Setup for the cancel I/O routine.
                            0218   806 ;
                            0218   807
                            0218   808 70$:
                    07      E1  0218   809          BBC     #CIN$V_CANCEL,-          ; Branch forward if no cancel
                07 08 AC        021A   810                  FLAGS(AP),80$            ; I/O routine was specified.
    00D0 C5   59    0C AB   C1  021D   811          ADDL3   CIN$L_CANCEL(R11),R9,-   ; Set up device cancel I/O
                            0224   812                  UCB$L_CI_CANCEL(R5)      ; routine address.
                            0224   813
                            0224   814 80$:
                00 BC   7D  0224   815          MOVQ    @BUFFER_DESC(AP),-       ; Store process-mapped buffer
            0090 C5         0227   816                  UCB$Q_CI_BUFDSC(R5)      ; descriptor too.
                            022A   817
                            022A   818 ;
                            022A   819 ; Allocate some blocks to be used as AST control blocks. The allocation
                            022A   820 ; raises IPL to IPL$_ASTDEL to prevent process deletion and subsequent
                            022A   821 ; loss of pool.
                            022A   822 ;
                            022A   823
                            022A   824 SETUP_ASTS:
                            022A   825
                            022A   826          .NOSHOW EXPANSIONS
                            022A   827
                            022A   828          ASSUME  UCB$W_CI_ACBNOW EQ UCB$W_CI_ACBCNT+2
        00A4 C5   D4        022A   829          CLRL    UCB$W_CI_ACBCNT(R5)      ; Note that no ACBs are needed
                            022E   830                                           ; or allocated at present.
                            022E   831
                            022E   832          .SHOW   EXPANSIONS
                            022E   833
        00AB C5   9E        022E   834          MOVAB   UCB$L_CI_AFLINK(R5),-    ; Initialize the UCB AST block
        00AB C5             0232   835                  UCB$L_CI_AFLINK(R5)      ; queue to point to itself.
        00AB C5   9E        0235   836          MOVAB   UCB$L_CI_AFLINK(R5),-    ; Ditto.
        00AC C5             0239   837                  UCB$L_CI_ABLINK(R5)
                09  B3      023C   838          BITW    #UCB$M_CI_EFN!UCB$M_CI_AST,- ; Efn or AST
                44  A5      023E   839                  UCB$L_DEVDEPEND(R5)      ; requested?
                4F  13      0240   840          BEQL    QUEUE_PACKET             ; Branch if not.
            50  14  3C      0242   841          MOVZWL  #SS$_BADPARAM,R0         ; Assume error in AST count.
        51  14  AC  D0      0245   842          MOVL    AST_COUNT(AP), R1        ; Get preallocated AST blocks count.
                10  13      0249   843          BEQL    20$                      ; Branch if paramenter absent.
            44 A5   04  C8  024B   844          BISL    #UCB$M_CI_REPEAT, -      ; Since count is present, set the
                            024F   845                  UCB$L_DEVDEPEND(R5)      ; repeat bit.
51  FFFF8000 8F    D3      024F   846          BITL    #^C^X7FFF, R1            ; Is count to big?
                05  13      0256   847          BEQL    30$                      ; Branch if count not to big.
                0041 31     0258   848          BRW     ERROR_DEALSPTS           ; Else, blow the request away.
                51  D6      025B   849 20$:     INCL    R1                       ; At least on AST block is needed.
                            025D   850 30$:
            50  1C  3C      025D   851          MOVZWL  #SS$_EXQUOTA,R0          ; Assume AST quota is too low.
```

```
                                0260   852          MOVL    UCB$L_CI_PCB(R5),R4     ; Restore PCB address.
                          38 A4 51  B1  0265   853          CMPW    R1,PCB$W_ASTCNT(R4)     ; Compare AST count with
                                0269   854                                           ; quota left.
                                03  15  0269   855          BLEQ    40$                     ; Branch forward if enough.
                              002E  31  026B   856          BRW     ERROR_DEALSPTS          ; Otherwise, stop with error.
                                026E   857
                                026E   858 ;
                                026E   859 ; Save the mode of the requesting mode in the UCB. Then allocate and
                                026E   860 ; initialize all the AST packets.
                                026E   861 ;
                                026E   862
                                026E   863 40$:
                                50  DC  026E   864          MOVPSL  R0                      ; Get the PSL.
                                16  EF  0270   865          EXTZV   #PSL$V_PRVMOD,-         ; Get process' mode from the
                           50   50  02      0272   866                  #PSL$S_PRVMOD,R0,R0     ; Get process' mode from PSL
                         0098 C5   50  90  0275   867          MOVB    R0,UCB$B_CI_ASTMOD(R5)  ; and store in the UCB.
                                027A   868
                                027A   869          .NOSHOW EXPANSIONS
                                027A   870
                                027A   871          ASSUME  AST_PARAMETER EQ AST_ROUTINE+4
                             0C AC  7D  027A   872          MOVQ    AST_ROUTINE(AP),-       ; Save the address of the AST
                           009C C5      027D   873                  UCB$L_CI_AST(R5)        ; routine and parameter in the
                                0280   874                                                  ; UCB.
                                0280   875
                                0280   876          .SHOW   EXPANSIONS
                                0280   877
                         00A4 C5   51  B0  0280   878          MOVW    R1,UCB$W_CI_ACBCNT(R5)  ; Save the number of ACBs
                                0285   879                                                  ; requested.
                                0038  30  0285   880          BSBW    CI_ALLOC_ASTS           ; Allocate and initialize all
                                0288   881                                                  ; AST control blocks.
                              06 50   E8  0288   882          BLBS    R0,QUEUE_PACKET         ; Branch forward on error.
                                028B   883
                                028B   884 ;
                                028B   885 ; If AST allocation and initialization failed, let it go unless the
                                028B   886 ; failure prevented even a single packet from being allocated. In the
                                028B   887 ; latter case, exit with error status from the connect.
                                028B   888 ;
                                028B   889
                           51   14 AC  D1  028B   890          CMPL    AST_COUNT(AP),R1        ; Any AST blocks allocated?
                                0B  13  028F   891          BEQL    ERROR_DEALSPTS          ; No. Exit with error.
                                0291   892
                                0291   893 ;
                                0291   894 ; Transfer control to an executive routine that queues the IRP or
                                0291   895 ; starts the driver in its start I/O routine. When the driver RSBs,
                                0291   896 ; the QIO completes by returning a success status to the process.
                                0291   897 ;
                                0291   898
                                0291   899 QUEUE_PACKET:                                  ; Queue packet to driver.
                         54   00B0 C5  D0  0291   900          MOVL    UCB$L_CI_PCB(R5),R4     ; Restore PCB address.
                          00000000'GF  17  0296   901          JMP     G^EXE$QIODRVPKT         ; Send packet to driver.
                                029C   902
                                029C   903 ;
                                029C   904 ; Error return. The instructions below assumes that an error status
                                029C   905 ; code is stored in R0.
                                029C   906 ;
                                029C   907 ; This outermost error condition happens after SPTs are allocated. The
                                029C   908 ; SPTs must be deallocated.
```

```
                                02B5    918
                                02B5    919 ;
                                02B5    920 ; This is a simple error. Just restore registers and return to caller
                                02B5    921 ; with status.
                                02B5    922 ;
                                02B5    923
                                02B5    924 ERROR:
54    00B0 C5    D0   02B5    925             MOVL    UCB$L_CI_PCB(R5),R4     ; Restore PCB address.
 00000000'GF     17   02BA    926             JMP     G^EXE$ABORTIO           ; Exit to QIO common code.
```

```
                     02C0   928          .SBTTL  CI_ALLOC_ASTS, Obtain and setup ASTs for process.
                     02C0   929
                     02C0   930 ;++
                     02C0   931 ; CI_ALLOC_ASTS - Set up some AST control blocks
                     02C0   932 ;
                     02C0   933 ; Functional description:
                     02C0   934 ;
                     02C0   935 ;       This routine gains control at IPL$_ASTDEL or at driver fork
                     02C0   936 ;       IPL.
                     02C0   937 ;
                     02C0   938 ;       This subroutine allocates and writes initial values into AST
                     02C0   939 ;       control blocks. Both the FDT routine and the driver fork process
                     02C0   940 ;       call this subroutine.
                     02C0   941 ;
                     02C0   942 ; Inputs:
                     02C0   943 ;
                     02C0   944 ;       R1      - number of AST control blocks to set up
                     02C0   945 ;       R4      - address of the process' PCB
                     02C0   946 ;       R5      - address of the UCB
                     02C0   947 ;
                     02C0   948 ; Implicit inputs:
                     02C0   949 ;
                     02C0   950 ;       UCB$L_CI_ABLINK - backward link into the UCB AST queue
                     02C0   951 ;       UCB$B_FIPL      - fork IPL of the driver
                     02C0   952 ;       PCB$W_ASTCNT    - number of ASTs left in process' quota
                     02C0   953 ;
                     02C0   954 ;       #ACB$K_LENGTH   - length of an ACB
                     02C0   955 ;       #DYN$C_ACB      - block type of an ACB
                     02C0   956 ;
                     02C0   957 ; Outputs:
                     02C0   958 ;
                     02C0   959 ;       R0      - status code:
                     02C0   960 ;
                     02C0   961 ;                       SS$_NORMAL      - success
                     02C0   962 ;                       SS$_INSFMEM     - insufficient nonpaged pool
                     02C0   963 ;
                     02C0   964 ;       R1      - number of blocks not allocated
                     02C0   965 ;       R2      - Contents destroyed
                     02C0   966 ;
                     02C0   967 ;       The subroutine preserves the contents of all other registers.
                     02C0   968 ;
                     02C0   969 ; Implicit outputs:
                     02C0   970 ;
                     02C0   971 ;       UCB$W_CI_ACBNOW records the number of ACBs currently allocated
                     02C0   972 ;       to the process.
                     02C0   973 ;
                     02C0   974 ;--
                     02C0   975
                     02C0   976 CI_ALLOC_ASTS:
0208 8F    BB        02C0   977          PUSHR   #^M<R3,R9>              ; Save volital registers
59   51    3C        02C4   978          MOVZWL  R1,R9                   ; Convert to long number blocks to get
                     02C7   979
                     02C7   980 ;
                     02C7   981 ; If quota permits, try to allocate another block. Exit on failure.
                     02C7   982 ;
                     02C7   983
                     02C7   984 LOOP:
```

```
                50    1C   3C   02C7   985          MOVZWL  #SS$_EXQUOTA,R0          ; Assume quota exhaustion error.
                  38 A4    B5   02CA   986          TSTW    PCB$W_ASTCNT(R4)        ; Any AST quota left?
                     2B    13   02CD   987          BEQL    10$                     ; No. Return with error.
                51    1C   D0   02CF   988          MOVL    #ACB$K_LENGTH,R1        ; Set up block size.
00000000'GF             16   02D2   989          JSB     G^EXE$ALONONPAGED       ; Allocate that block.
                   1F 50    E9   02D8   990          BLBC    R0,10$                  ; Branch forward if error.
                                02DB   991
                                02DB   992 ;
                                02DB   993 ; A block is allocated. Decrement quota; increment count allocated in
                                02DB   994 ; the UCB, link the block into the ACB queue, and initialize the block.
                                02DB   995 ;
                                02DB   996
                  38 A4    B7   02DB   997          DECW    PCB$W_ASTCNT(R4)        ; Decrement AST quota.
             08 A2    51   B0   02DE   998          MOVW    R1,ACB$W_SIZE(R2)       ; Set size of block allocated
                      02   90   02E2   999          MOVB    #DYN$C_ACB,-            ; Load ACB type field
                   0A A2        02E4  1000                  ACB$B_TYPE(R2)
                   0B A5    90   02E6  1001          MOVB    UCB$B_FIPL(R5),-        ; Load fork IPL
                   0B A2        02E9  1002                  ACB$B_RMOD(R2)
                      62    0E   02EB  1003          INSQUE  ACB$L_ASTQFL(R2),-      ; Insert new ACB in the queue
                00AC D5          02ED  1004                  @UCB$L_CI_ABLINK(R5)    ;
                00A6 C5    B6   02F0  1005          INCW    UCB$W_CI_ACBNOW(R5)     ; Increment number allocated
                                02F4  1006
                                02F4  1007 ;
                                02F4  1008 ; See if more blocks to initialize. If not, just return to caller.
                                02F4  1009 ;
                                02F4  1010
                  D0 59    F5   02F4  1011          SOBGTR  R9,LOOP                 ; Loop back if not done yet.
                50    01   3C   02F7  1012          MOVZWL  #SS$_NORMAL,R0          ; Set up success status code.
                                02FA  1013
                                02FA  1014 10$:
                51    59   D0   02FA  1015          MOVL    R9,R1                   ; Restore number of blocks left
                0208 8F    BA   02FD  1016          POPR    #^M<R3,R9>              ; Restore saved registers
                           05   0301  1017          RSB                             ; Return.
```

```
                     0302  1019          .SBTTL  CI_START, Start I/O routine
                     0302  1020
                     0302  1021 ;++
                     0302  1022 ; CI_START - Start the device.
                     0302  1023 ;
                     0302  1024 ; Functional description:
                     0302  1025 ;
                     0302  1026 ;       When this routine gains control, IPL is at driver fork level.
                     0302  1027 ;
                     0302  1028 ;       This routine obtains the address of an argument list from the
                     0302  1029 ;       UCB, and then JSBs to a user-specified start device routine.
                     0302  1030 ;       If the user requested a CALL interface, the JSB transfers
                     0302  1031 ;       control to the label CI_START_CALL (in this routine), which
                     0302  1032 ;       actually executes the CALLG to the user-specified routine.
                     0302  1033 ;
                     0302  1034 ;       When the user routine is called, the following inputs apply:
                     0302  1035 ;
                     0302  1036 ;               R2      - points to counted argument list
                     0302  1037 ;               R3      - address of the IRP
                     0302  1038 ;               R5      - address of the UCB
                     0302  1039 ;
                     0302  1040 ;               the counted argument list is as follows:
                     0302  1041 ;
                     0302  1042 ;                0(R2)  - the argument count (4)
                     0302  1043 ;                4(R2)  - the system-mapped user buffer address
                     0302  1044 ;                8(R2)  - the IRP address
                     0302  1045 ;               12(R2)  - the system-mapped address of the device's CSR
                     0302  1046 ;               16(R2)  - the UCB address
                     0302  1047 ;
                     0302  1048 ; Inputs:
                     0302  1049 ;
                     0302  1050 ;       R3      - address of the IRP (I/O request packet)
                     0302  1051 ;       R5      - address of the UCB (unit control block)
                     0302  1052 ;
                     0302  1053 ; Implicit inputs:
                     0302  1054 ;
                     0302  1055 ;       The prepared argument list for a CALLG is at UCB$L_CI_STARGC.
                     0302  1056 ;
                     0302  1057 ;       The address of the user-specified start device routine needing
                     0302  1058 ;       a CALL interface is at UCB$L_CI_STACAL.
                     0302  1059 ;
                     0302  1060 ; Outputs:
                     0302  1061 ;
                     0302  1062 ;       R0      - 1st longword of I/O status: contains status code and
                     0302  1063 ;                 number of bytes transferred
                     0302  1064 ;       R1      - 2nd longword of I/O status: device-dependent
                     0302  1065 ;
                     0302  1066 ;       The routine must preserve all registers except R0-R2 and R4.
                     0302  1067 ;
                     0302  1068 ;--
                     0302  1069
                     0302  1070 CI_START:                                     ; Start the device.
52   00D4 C5   9E    0302  1071          MOVAB   UCB$L_CI_STARGC(R5),R2      ; Get address of argument block.
     00C0 D5   16    0307  1072          JSB     @UCB$L_CI_START(R5)         ; JSB indirect through UCB to
                     030B  1073                                              ; a start device routine.
               05    030B  1074          RSB                                 ; Then return.
                     030C  1075
```

```
                     030C  1076 ;
                     030C  1077 ; Use the CALL interface.
                     030C  1078 ;
                     030C  1079
                     030C  1080 CI_START_CALL:
          62    FA   030C  1081          CALLG   (R2),-                  ; Call the user's start device
00C4 D5              030E  1082                  @UCB$L_CI_STACAL(R5)    ; routine.
                     0311  1083                                          ; routine.
                05   0311  1084          RSB                             ; Return.
```

```
0312  1086          .SBTTL  CI_INTERRUPT, Interrupt service routine
0312  1087
0312  1088 ;++
0312  1089 ; CI_INTERRUPT, Analyzes interrupts, processes solicited interrupts
0312  1090 ;
0312  1091 ; Functional description:
0312  1092 ;
0312  1093 ;       When this routine gains control, IPL is at device fork level.
0312  1094 ;
0312  1095 ;       This routine obtains the address of an argument list from the
0312  1096 ;       UCB, and then JSBs to a user-specified interrupt service
0312  1097 ;       routine. If the user requested a CALL interface, the JSB
0312  1098 ;       transfers control to the label CI_ISR_CALL (in this routine),
0312  1099 ;       which actually executes the CALLG to the user-specified routine.
0312  1100 ;
0312  1101 ;       When the user's interrupt service routine gains control, the
0312  1102 ;       following inputs apply:
0312  1103 ;
0312  1104 ;               R2      - address of counted argument list
0312  1105 ;               R4      - address of the IDB
0312  1106 ;               R5      - address of the UCB
0312  1107 ;
0312  1108 ;               the counted argument list is as follows:
0312  1109 ;
0312  1110 ;                0(R2)  - count of arguments (5)
0312  1111 ;                4(R2)  - the system-mapped address of the user buffer
0312  1112 ;                8(R2)  - the address of the AST parameter
0312  1113 ;               12(R2)  - the system-mapped address of the device's CSR
0312  1114 ;               16(R2)  - the address of the IDB
0312  1115 ;               20(R2)  - the address of the UCB
0312  1116 ;
0312  1117 ;       When the user's interrupt service routine returns, this ISR
0312  1118 ;       checks the status code in R0. A success status results in the
0312  1119 ;       creation of a fork process to set an event flag or queue an AST
0312  1120 ;       to the process. A low-bit-clear status causes immediate
0312  1121 ;       dismissal of the interrupt.
0312  1122 ;
0312  1123 ;       The fork block queued is either an ACB from the queue in the
0312  1124 ;       UCB, or the UCB itself. In the latter case, a bit is set to
0312  1125 ;       force a disconnect from the interrupt since no ACBs are left to
0312  1126 ;       permit further forking or further AST queuing.
0312  1127 ;
0312  1128 ;       The fork process is described further below.
0312  1129 ;
0312  1130 ; Inputs:
0312  1131 ;
0312  1132 ;        0(SP)  - pointer to the address of the IDB (interrupt data
0312  1133 ;                 block)
0312  1134 ;        4(SP)  - saved R0
0312  1135 ;        8(SP)  - saved R1
0312  1136 ;       12(SP)  - saved R2
0312  1137 ;       16(SP)  - saved R3
0312  1138 ;       20(SP)  - saved R4
0312  1139 ;       24(SP)  - saved R5
0312  1140 ;       28(SP)  - saved PSL (program status longword)
0312  1141 ;       32(SP)  - saved PC
0312  1142 ;
```

```
                     0312  1143 ;       The IDB contains the CSR address and the UCB address.
                     0312  1144 ;
                     0312  1145 ; Implicit inputs:
                     0312  1146 ;
                     0312  1147 ;       The prepared argument list for a CALLG is at UCB$L_CI_ISARGC.
                     0312  1148 ;
                     0312  1149 ;       The address of the user-specified interrupt service routine
                     0312  1150 ;       needing a CALL interface is at UCB$L_ISRCAL.
                     0312  1151 ;
                     0312  1152 ; Outputs:
                     0312  1153 ;
                     0312  1154 ;       The routine must preserve all registers except R0-R5.
                     0312  1155 ;
                     0312  1156 ;--
                     0312  1157
                     0312  1158 CI_INTERRUPT:                                ; Service device interrupt
          54   9E   D0  0312  1159         MOVL    @(SP)+,R4                 ; Get address of IDB and remove
                     0315  1160                                             ; pointer from stack.
     55   04 A4   D0  0315  1161         MOVL    IDB$L_OWNER(R4),R5        ; Get address of device owner's
                     0319  1162                                             ; UCB.
52   00E8 C5   9E  0319  1163         MOVAB   UCB$L_CI_ISARGC(R5),R2    ; Get argument list address.
     00C8 D5   16  031E  1164         JSB     @UCB$L_CI_ISR(R5)         ; JSB to user-routine.
       09 50   E8  0322  1165         BLBS    R0,CHECK_AST              ; Branch to fork on success.
                     0325  1166
                     0325  1167 ;
                     0325  1168 ; Restore registers and dismiss the interrupt.
                     0325  1169 ;
                     0325  1170
                     0325  1171 DISMISS_INT:
          3F   BA   0325  1172         POPR    #^M<R0,R1,R2,R3,R4,R5>    ; Restore 6 registers.
               02   0327  1173         REI                               ; Return from interrupt.
                     0328  1174
                     0328  1175 ;
                     0328  1176 ; Use the CALL interface. The return is to the JSB 5 lines earlier.
                     0328  1177 ;
                     0328  1178
                     0328  1179 CI_ISR_CALL:
          62   FA   0328  1180         CALLG   (R2),-                    ; Call the user's ISR.
     00CC D5           032A  1181                 @UCB$L_CI_ISRCAL(R5)
               05   032D  1182         RSB                               ; Return to JSB caller above.
                     032E  1183
                     032E  1184 ;
                     032E  1185 ; See whether an AST delivery is required.
                     032E  1186 ;
                     032E  1187
                     032E  1188 CHECK_AST:
          09   B3   032E  1189         BITW    #UCB$M_CI_AST!UCB$M_CI_EFN,- ; AST or efn requested?
       44 A5           0330  1190                 UCB$L_DEVDEPEND(R5)
          F1   13   0332  1191         BEQL    DISMISS_INT               ; Branch if not.
                     0334  1192
                     0334  1193 10$:
          53   55   D0  0334  1194         MOVL    R5,R3                     ; Save UCB address.
55   00A8 D3   0F  0337  1195         REMQUE  @UCB$L_CI_AFLINK(R3),R5   ; Get the address of an ACB.
          08   1C   033C  1196         BVC     20$                       ; If ACB found, branch forward.
     55   53   D0  033E  1197         MOVL    R3,R5                     ; Restore UCB address to R5.
          08   E2   0341  1198         BBSS    #UCB$V_CI_UCBFRK,-        ; Set the "forking on UCB" bit
       44 A5       0343  1199                 UCB$L_DEVDEPEND(R5),-     ; in UCB, and, if already set,
```

```
                     DF        0345  1200                    DISMISS_INT             ; just go dismiss the interrupt.
                               0346  1201
                               0346  1202 ;
                               0346  1203 ; Create the fork process.
                               0346  1204 ;
                               0346  1205
                               0346  1206 20$:
                        DC AF  9F    0346  1207             PUSHAB  DISMISS_INT     ; Put a return address on stack.
                               0349  1208             FORK                        ; Create a fork process.
                  00000000'GF  16    0349                    JSB     G^EXE$FORK
                               034F
```

```
                        034F  1210          .SBTTL  CI_FORK_PROCESS - Queues ASTs and sets event flags
                        034F  1211
                        034F  1212 ;++
                        034F  1213 ; CI_FORK_PROCESS - Fork process created after an interrupt
                        034F  1214 ;
                        034F  1215 ; Functional description:
                        034F  1216 ;
                        034F  1217 ;       The fork process, according to flag settings in the UCB, queues
                        034F  1218 ;       an AST to the process, sets an event flag for the process,
                        034F  1219 ;       replenishes the ACB supply to anticipate future interrupts,
                        034F  1220 ;       and, in the event of errors, disconnects the device from the
                        034F  1221 ;       process.
                        034F  1222 ;
                        034F  1223 ; Inputs:
                        034F  1224 ;
                        034F  1225 ;       R3      - address of the UCB
                        034F  1226 ;       R5      - address of the AST/fork control block
                        034F  1227 ;
                        034F  1228 ; Outputs:
                        034F  1229 ;
                        034F  1230 ;       The routine may destroy R0-R5, but must preserve all other
                        034F  1231 ;       registers.
                        034F  1232 ;
                        034F  1233 ;       In the event of an error, this routine sets up the following
                        034F  1234 ;       registers and branches into the cancel I/O code:
                        034F  1235 ;
                        034F  1236 ;               R3      - address of the IRP
                        034F  1237 ;               R4      - address of the PCB
                        034F  1238 ;               R5      - address of the UCB
                        034F  1239 ;
                        034F  1240 ;--
                        034F  1241
                        034F  1242 CI_FORK_PROCESS:
54   00B0 C3   D0       034F  1243          MOVL    UCB$L_CI_PCB(R3),R4     ; Get address of owner PCB.
        10   91         0354  1244          CMPB    #DYN$C_UCB,-            ; Is this a UCB fork block?
          0A A5         0356  1245                  ACB$B_TYPE(R5)
             03   12    0358  1246          BNEQ    10$                     ; Branch if not.
           007D   31    035A  1247          BRW     70$                     ; Else go disconnect device
                        035D  1248                                          ; from process
                        035D  1249 10$:
             03   E1    035D  1250          BBC     #UCB$V_CI_AST,-         ; If no AST needs queuing,
        2F 44 A3         035F  1251                  UCB$L_DEVDEPEND(R3),20$ ; just branch forward.
                        0362  1252
                        0362  1253 ;
                        0362  1254 ; Set up the AST control block and queue the AST to the process.
                        0362  1255 ;
                        0362  1256
         52   01   D0    0362  1257          MOVL    #PRI$_IOCOM,R2          ; Set priority increment class.
       0098 C3   90      0365  1258          MOVB    UCB$B_CI_ASTMOD(R3),-   ; Load AST delivery mode into
          0B A5         0369  1259                  ACB$B_RMOD(R5)          ; AST block.
          40 8F   88    036B  1260          BISB    #ACB$M_QUOTA,-          ; Set the bit that causes AST
          0B A5         036E  1261                  ACB$B_RMOD(R5)          ; delivery code to return quota.
          60 A4   D0    0370  1262          MOVL    PCB$L_PID(R4),-         ; Store PID in the AST block.
          0C A5         0373  1263                  ACB$L_PID(R5)
                        0375  1264
                        0375  1265          .NOSHOW EXPANSIONS
                        0375  1266
```

```
                     0375  1267          ASSUME  UCB$L_CI_ASTPRM EQ UCB$L_CI_AST+4
                     0375  1268          ASSUME  ACB$L_ASTPRM EQ ACB$L_AST+4
          009C C3 7D 0375  1269          MOVQ    UCB$L_CI_AST(R3),-      ; Store AST routine address and
            10 A5    0379  1270                  ACB$L_AST(R5)           ; parameter.
                     037B  1271
                     037B  1272          .SHOW   EXPANSIONS
                     037B  1273
                  18 BB 037B  1274          PUSHR   #^M<R3,R4>              ; Save UCB and PCB addresses.
00000000'GF       16    037D  1275          JSB     G^SCH$QAST              ; Queue the AST to the process.
                  18 BA 0383  1276          POPR    #^M<R3,R4>              ; Restore UCB and PCB addresses.
               05 50    E8  0385  1277          BLBS    R0,15$                  ; Branch forward on success.
                     0388  1278
                     0388  1279 ;
                     0388  1280 ; AST QUEUING FAILED.  DISCONNECT DEVICE FROM PROCESS.
                     0388  1281 ;
                     0388  1282
               55 53 D0    0388  1283          MOVL    R3,R5                   ; Load UCB address into R5.
                  5A 11    038B  1284          BRB     IO_COMPLETE             ; Go disconnect device.
                     038D  1285
                     038D  1286 15$:                                        ; An AST was actually queued.
          00A6 C3 B7  038D  1287          DECW    UCB$W_CI_ACBNOW(R3)     ; Decrement current ACB count.
                     0391  1288
                     0391  1289 ;
                     0391  1290 ; If an event flag was specified, post the event flag.
                     0391  1291 ;
                     0391  1292
                     0391  1293 20$:
                  55 DD    0391  1294          PUSHL   R5                      ; Save fork block address.
               55 53 D0    0393  1295          MOVL    R3,R5                   ; Move UCB address into R5.
                  00 E1    0396  1296          BBC     #UCB$V_CI_EFN,-         ; Any event flag specified?
            17 44 A5       0398  1297                  UCB$L_DEVDEPEND(R5),30$ ; Branch forward if none.
               52 01 D0    039B  1298          MOVL    #PRI$_IOCOM,R2          ; Set priority increment class.
         51    60 A4 D0    039E  1299          MOVL    PCB$L_PID(R4),R1        ; Get PID address.
      53    009A C5 3C     03A2  1300          MOVZWL  UCB$W_CI_EFNUM(R5),R3   ; Get event flag number.
00000000'GF       16    03A7  1301          JSB     G^SCH$POSTEF            ; Go set the event flag.
               02 50    E8  03AD  1302          BLBS    R0,30$                  ; Branch if efn post succeeded
                  32 11    03B0  1303          BRB     90$                     ; Else disconnect process.
                     03B2  1304
                     03B2  1305 ;
                     03B2  1306 ; If the user only asked for a single AST delivery or a single
                     03B2  1307 ; interrupt, go disconnect the device from the process, and thus
                     03B2  1308 ; complete the connect to interrupt I/O request.
                     03B2  1309 ;
                     03B2  1310
                     03B2  1311 30$:
                  02 E1    03B2  1312          BBC     #UCB$V_CI_REPEAT,-      ; Branch if user specified
            2A 44 A5       03B4  1313                  UCB$L_DEVDEPEND(R5),80$ ; only one AST/event flag
                     03B7  1314                                                 ; be delivered.
               50 8ED0     03B7  1315          POPL    R0                      ; Restore fork block addr.
                     03BA  1316
                     03BA  1317 ;
                     03BA  1318 ; If the AST was queued to the process, then go ahead and allocate
                     03BA  1319 ; a replacement block. Otherwise, relink the ACB used as a fork block
                     03BA  1320 ; back into the UCB ACB queue.
                     03BA  1321 ;
                     03BA  1322
                  03 E0    03BA  1323          BBS     #UCB$V_CI_AST,-         ; Branch forward if an AST
```

```
                06 44 A5        03BC  1324                UCB$L_DEVDEPEND(R5),50$ ; was queued.
                      60    0E  03BF  1325          INSQUE  ACB$L_ASTQFL(R0),-      ; Otherwise, relink the ACB
                00AC D5         03C1  1326                  @UCB$L_CI_ABLINK(R5)    ; back into the UCB queue.
                              05  03C4  1327          RSB                             ; And exit from fork process.
                                  03C5  1328
                                  03C5  1329 ;
                                  03C5  1330 ; Replenish the number of available ACBs, and initialize them. If no
                                  03C5  1331 ; pool is available, let the replenishment happen on the next interrupt.
                                  03C5  1332 ; If no ACBs are left, the next interrupt will force an I/O completion
                                  03C5  1333 ; because only one fork on the UCB is possible.
                                  03C5  1334 ;
                                  03C5  1335
                                  03C5  1336 50$:
                00A6 C5     A3    03C5  1337          SUBW3   UCB$W_CI_ACBNOW(R5),-   ; See how many ACBs need to be
           51   00A4 C5           03C9  1338                  UCB$W_CI_ACBCNT(R5),R1  ; allocated.
                  FEF0      30    03CD  1339          BSBW    CI_ALLOC_ASTS           ; Initialize the blocks.
                    06 50   E8    03D0  1340          BLBS    R0,60$                  ; Branch forward on success.
                                  03D3  1341
                                  03D3  1342 ;
                                  03D3  1343 ; Some failure occurred in attempting to replenish the ACBs. If no ACBs
                                  03D3  1344 ; are currently allocated, disconnect the device from the process
                                  03D3  1345 ; because no other interrupts can be handled.
                                  03D3  1346 ;
                                  03D3  1347
                00A6 C5     B5    03D3  1348          TSTW    UCB$W_CI_ACBNOW(R5)     ; Any ACBs allocated?
                      01    13    03D7  1349          BEQL    70$                     ; No. Disconnect the process.
                                  03D9  1350
                                  03D9  1351 60$:
                              05  03D9  1352          RSB                             ; Return.
                                  03DA  1353
                                  03DA  1354 ;
                                  03DA  1355 ; The UCB was used as a fork block. Load the disconnect error code into
                                  03DA  1356 ; R0 before disconnecting the process.
                                  03DA  1357 ;
                                  03DA  1358
                                  03DA  1359 70$:
           50   204C 8F     3C    03DA  1360          MOVZWL  #SS$_DISCONNECT,R0      ; Setup status code.
                      06    11    03DF  1361          BRB     IO_COMPLETE             ; Complete disconnect.
                                  03E1  1362
                                  03E1  1363 ;
                                  03E1  1364 ; Only a single AST or event flag was requested.  Set status
                                  03E1  1365 ; to success, clean stack, and disconnect.
                                  03E1  1366 ;
                                  03E1  1367
                50    01    3C    03E1  1368 80$:    MOVZWL  #SS$_NORMAL,R0          ; Set status to success.
                                  03E4  1369
                                  03E4  1370 ;
                                  03E4  1371 ; Event flag posting failed.  Status is in R0.  Clear stack,
                                  03E4  1372 ; and disconnect.
                                  03E4  1373 ;
                                  03E4  1374
                      54 8ED0     03E4  1375 90$:    POPL    R4                      ; Clear stack of fork blk
                                  03E7  1376                                         ; address
                                  03E7  1377
                                  03E7  1378 ;
                                  03E7  1379 ; Complete the I/O, thereby disconnecting the process from the device.
                                  03E7  1380 ; This is necessary if the UCB was used as a fork block to prevent
```

```
                     03E7  1381 ; the single UCB from being used many times simultaneously as a fork
                     03E7  1382 ; block.
                     03E7  1383
                     03E7  1384 IO_COMPLETE:
54   00B0 C5   D0    03E7  1385          MOVL    UCB$L_CI_PCB(R5),R4     ; Set up PCB address.
  53   58 A5   D0    03EC  1386          MOVL    UCB$L_IRP(R5),R3        ; Set up IRP address.
           0B   11    03F0  1387          BRB     CI_FORCE_CANCEL         ; Fall through to join the
                     03F2  1388                                          ; cancel I/O code.
```

```
                                                B 1
                        .SBTTL  CI_CANCEL, Cancel I/O routine

                ;++
                ; CI_CANCEL, Cancels an I/O operation in progress
                ;
                ; Functional description:
                ;
                ;       When this routine gains control, IPL is at driver fork level.
                ;
                ;       This routine calls IOC$CANCELIO to set the cancel bit in the
                ;       UCB status word if:
                ;
                ;               the device is busy,
                ;               the IRP's process ID matches the cancel process ID,
                ;               the IRP channel matches the cancel channel.
                ;
                ;       If IOC$CANCELIO sets the cancel bit, then this driver routine
                ;       calls a user-specified cancel I/O routine. The call interface
                ;       is JSB or CALLS depending on a bit setting in the UCB. On
                ;       entry to the user routine, the register settings are unchanged.
                ;       For the CALL interface, the argument list is as follows:
                ;
                ;                0(AP)  - argument count
                ;                4(AP)  - negated value of the channel index number
                ;                8(AP)  - address of the IRP (I/O request packet)
                ;               12(AP)  - address of the PCB (process control block) for
                ;                         the process canceling I/O
                ;               16(AP)  - address of the UCB (unit control block)
                ;
                ; Inputs:
                ;
                ;       R2      - negated value of the channel index number
                ;       R3      - address of the current IRP (I/O request packet)
                ;       R4      - address of the PCB (process control block) for the
                ;                 process canceling I/O
                ;       R5      - address of the UCB (unit control block)
                ;
                ; Implicit inputs:
                ;
                ;       UCB$V_CI_USECAL is set in UCB$L_DEVDEPEND if the CALLS
                ;       interface was requested.
                ;
                ; Outputs:
                ;
                ;       The routine must preserve all registers except R0-R3.
                ;
                ;       The routine may set the UCB$M_CANCEL bit in UCB$W_STS.
                ;
                ;--

                CI_CANCEL:                              ; Cancel an I/O operation
00000000'GF   16        JSB     G^IOC$CANCELIO          ; Set cancel bit if appropriate.
      03   E1           BBC     #UCB$V_CANCEL,-         ; If the cancel bit is not set,
    64   A5                      UCB$W_STS(R5),-         ; just return.
      24                        CANCEL_EXIT

                ;
```

```
                     03FD  1447 ; Device-dependent cancel operations go next.
                     03FD  1448 ;
                     03FD  1449
                     03FD  1450 CI_FORCE_CANCEL:
            08   E1  03FD  1451         BBC     #UCB$V_BSY,-                    ; Branch forward if device does
          1D 64 A5   03FF  1452                 UCB$W_STS(R5),20$               ; not have IRP associated.
            01   E1  0402  1453         BBC     #UCB$V_CI_USECAL,-              ; Branch to JSB code if user
          14 44 A5   0404  1454                 UCB$L_DEVDEPEND(R5),10$         ; didn't request CALL interface.
       0F 44 A5  07   E1  0407  1455         BBC     #UCB$V_CI_CANCEL,-          ; Branch to JSB code if user
                     040C  1456                 UCB$L_DEVDEPEND(R5), 10$;       cancel routine doesn't exist.
                     040C  1457
                     040C  1458 ;
                     040C  1459 ; Load the input registers onto the argument stack and CALLS the
                     040C  1460 ; user-specified cancel I/O routine.
                     040C  1461 ;
                     040C  1462
            55   DD  040C  1463         PUSHL   R5                              ; Push address of UCB.
            54   DD  040E  1464         PUSHL   R4                              ; Push address of PCB.
            53   DD  0410  1465         PUSHL   R3                              ; Push address of IRP.
            52   DD  0412  1466         PUSHL   R2                              ; Push negated channel index.
      00D0 D5   04   FB  0414  1467         CALLS   #4,@UCB$L_CI_CANCEL(R5) ; Call user's cancel I/O
                     0419  1468                                                 ; routine.
            04   11  0419  1469         BRB     20$                             ; Go disconnect device.
                     041B  1470
                     041B  1471 ;
                     041B  1472 ; Just JSB to the user-specified cancel I/O routine.
                     041B  1473 ;
                     041B  1474
                     041B  1475 10$:                                            ; JSB path.
      00D0 D5   16   041B  1476         JSB     @UCB$L_CI_CANCEL(R5)            ; JSB to user's cancel I/O
                     041F  1477                                                 ; routine.
                     041F  1478
                     041F  1479 ;
                     041F  1480 ; Now disconnect the process from the interrupt by restoring the dummy
                     041F  1481 ; device handling routine addresses and completing the I/O.
                     041F  1482 ;
                     041F  1483
                     041F  1484 20$:
            01   10  041F  1485         BSBB    CI_DISCONNECT                   ; Disconnect device from
                     0421  1486                                                 ; process.
                     0421  1487
                     0421  1488 ;
                     0421  1489 ; A simple return if the cancel does not apply.
                     0421  1490 ;
                     0421  1491
                     0421  1492 CANCEL_EXIT:
            05   0421  1493         RSB                                     ; Return.
```

```
                     0422  1495          .SBTTL  CI_DISCONNECT, Disconnect the process from the device
                     0422  1496
                     0422  1497 ;++
                     0422  1498 ; CI_DISCONNECT, Restores the device to a null-driver state
                     0422  1499 ;
                     0422  1500 ; Functional description:
                     0422  1501 ;
                     0422  1502 ;       When this routine gains control, IPL is at driver fork level.
                     0422  1503 ;
                     0422  1504 ;       This subroutine performs a disconnect in the following steps:
                     0422  1505 ;
                     0422  1506 ;               Restores the dummy routine address to the four
                     0422  1507 ;                       possible process-supplied kernel mode routines
                     0422  1508 ;               Deallocates the realtime SPTs reserved to the process.
                     0422  1509 ;               Deallocates unused AST control blocks
                     0422  1510 ;               Completes the QIO request, if one is outstanding
                     0422  1511 ;
                     0422  1512 ; Inputs:
                     0422  1513 ;
                     0422  1514 ;       R0      - I/O completion status from user's cancel routine
                     0422  1515 ;       R1      - more completion status
                     0422  1516 ;       R4      - address of the process' PCB
                     0422  1517 ;       R5      - address of the device's UCB
                     0422  1518 ;
                     0422  1519 ; Outputs:
                     0422  1520 ;
                     0422  1521 ;       The routine preserves all registers.
                     0422  1522 ;
                     0422  1523 ;--
                     0422  1524
                     0422  1525 CI_DISCONNECT:
         0F   BB     0422  1526          PUSHR   #^M<R0,R1,R2,R3>        ; Save registers.
       44 A5  B4     0424  1527          CLRW    UCB$L_DEVDEPEND(R5)     ; Clear the flags word.
00000482'EF   DE     0427  1528          MOVAL   CI_DUMMY_RSB,-          ; Restore dummy device
       00BC C5       042D  1529                  UCB$L_CI_INIDEV(R5)     ; initialization routine addr.
00000482'EF   DE     0430  1530          MOVAL   CI_DUMMY_RSB,-          ; Restore dummy start device
       00C0 C5       0436  1531                  UCB$L_CI_START(R5)      ; routine address.
00000482'EF   DE     0439  1532          MOVAL   CI_DUMMY_RSB,-          ; Restore dummy interrupt
       00C8 C5       043F  1533                  UCB$L_CI_ISR(R5)        ; service routine address.
00000482'EF   DE     0442  1534          MOVAL   CI_DUMMY_RSB,-          ; Restore dummy cancel I/O
       00D0 C5       0448  1535                  UCB$L_CI_CANCEL(R5)     ; routine address.
                     044B  1536
                     044B  1537 ;
                     044B  1538 ; Deallocate the SPTs that are double mapping the user buffer in
                     044B  1539 ; system address space.
                     044B  1540 ;
                     044B  1541
   52    00B4 C5     044B  1542          MOVAQ   UCB$Q_CI_SPTDSC(R5),R2  ; Get SPT descriptor.
         62   D5     0450  1543          TSTL    CIN$L_SPTCOUNT(R2)      ; Any allocated?
         0A   13     0452  1544          BEQL    10$                     ; No. Branch forward.
00000542'GF   16     0454  1545          JSB     G^EXE$DEAL_SPTS         ; Yes. Deallocate them.
       00B4 C5       045A  1546          CLRQ    UCB$Q_CI_SPTDSC(R5)     ; Clear out SPT descriptor.
                     045E  1547
                     045E  1548 ;
                     045E  1549 ; For each AST control block in the UCB queue, deallocate the space.
                     045E  1550 ; Then restore process quota for these blocks.
                     045E  1551 ;
```

```
                     045E  1552
                     045E  1553 10$:
50   00A8 D5   0F    045E  1554          REMQUE  @UCB$L_CI_AFLINK(R5),R0 ; Get the address of an AST
                     0463  1555                                          ; control block.
          0F   1D    0463  1556          BVS     20$                     ; Branch if no more exist.
00000000'GF   16    0465  1557          JSB     G^EXE$DEANONPAGED       ; Deallocate the block.
        38 A4 B6    046B  1558          INCW    PCB$W_ASTCNT(R4)        ; Increment AST quota.
      00A6 C5 B7    046E  1559          DECW    UCB$W_CI_ACBNOW(R5)     ; Decrement ACBs allocated.
          EA   11   0472  1560          BRB     10$                     ; Go look for another.
                     0474  1561
                     0474  1562 ;
                     0474  1563 ; Check the UCB to see if the device has an IRP associated with it.
                     0474  1564 ; If not, just return. Otherwise, complete the I/O request by a
                     0474  1565 ; transfer of control to VMS. The I/O completion disconnects the
                     0474  1566 ; process from the interrupt.
                     0474  1567 ;
                     0474  1568
                     0474  1569 20$:
          0F   BA    0474  1570          POPR    #^M<R0,R1,R2,R3>        ; Restore I/O status.
          08   E0    0476  1571          BBS     #UCB$V_BSY,-            ; Branch forward if device is
    01 64 A5          0478  1572                  UCB$W_STS(R5),30$       ; connected to a process.
               05     047B  1573          RSB                             ; Otherwise, just return.
                     047C  1574
                     047C  1575 30$:
                     047C  1576          REQCOM                          ; Complete the I/O.
00C00000'GF   17    047C                        JMP     G^IOC$REQCOM
                     0482
```

```
                     0482  1578          .SBTTL  CI_DUMMY_RSB
                     0482  1579
                     0482  1580 ;++
                     0482  1581 ; CI_DUMMY_RSB - nop routine
                     0482  1582 ;
                     0482  1583 ; Functional description:
                     0482  1584 ;
                     0482  1585 ;       This routine returns to caller with a RSB instruction.
                     0482  1586 ;
                     0482  1587 ; Inputs:
                     0482  1588 ;
                     0482  1589 ;       none
                     0482  1590 ;
                     0482  1591 ; Outputs:
                     0482  1592 ;
                     0482  1593 ;       R0 contains the SS$_NORMAL status code.
                     0482  1594 ;
                     0482  1595 ;--
                     0482  1596
                     0482  1597 CI_DUMMY_RSB:
 50   01   3C       0482  1598          MOVZWL  #SS$_NORMAL,R0          ; Load success status.
                     05  0485  1599          RSB                             ; Return.
```

```
                        0486  1601          .SBTTL  EXE$ALLOC_SPTS, Allocate a contiguous set of SPTs
                        0486  1602
                        0486  1603 ;++
                        0486  1604 ; EXE$ALLOC_SPTS - Allocate SPTs to double map the user's buffer
                        0486  1605 ;
                        0486  1606 ; Functional description:
                        0486  1607 ;
                        0486  1608 ;       When this routine gains control, IPL is at driver fork level.
                        0486  1609 ;
                        0486  1610 ;       Using a bit map whose address is stored in the control block
                        0486  1611 ;       addressed by EXE$GL_RTBITMAP, try to allocate "n" contiguous
                        0486  1612 ;       SPTs.
                        0486  1613 ;
                        0486  1614 ; Inputs:
                        0486  1615 ;
                        0486  1616 ;       R2      - address of a quadword descriptor:
                        0486  1617 ;
                        0486  1618 ;                         CIN$L_SPTCOUNT(R2)      - count of SPTs needed
                        0486  1619 ;                         CIN$L_STARTVPN(R2)      - zero
                        0486  1620 ;
                        0486  1621 ; Implicit inputs:
                        0486  1622 ;
                        0486  1623 ;       EXE$GL_RTBITMAP - address of SPT bit map control block
                        0486  1624 ;
                        0486  1625 ;                 ----------------------------------
                        0486  1626 ;                 !          starting VPN          !
                        0486  1627 ;                 ----------------------------------
                        0486  1628 ;                 !       number of SPTs left      !
                        0486  1629 ;                 ----------------------------------
                        0486  1630 ;                 !       !  type  !      size     !
                        0486  1631 ;                 ----------------------------------
                        0486  1632 ;                 !                                !
                        0486  1633 ;                 !                                !
                        0486  1634 ;                 !             bitmap             !
                        0486  1635 ;                 !                                !
                        0486  1636 ;                 ----------------------------------
                        0486  1637 ; Outputs:
                        0486  1638 ;
                        0486  1639 ;       R0      - status code:
                        0486  1640 ;
                        0486  1641 ;                         SS$_NORMAL      - success
                        0486  1642 ;                         SS$_INSFSPTS    - not enough contiguous SPTs
                        0486  1643 ;
                        0486  1644 ;       R2      - address of the quadword descriptor:
                        0486  1645 ;
                        0486  1646 ;                         0(R2)   - count of SPTs allocated
                        0486  1647 ;                         4(R2)   - starting VPN
                        0486  1648 ;
                        0486  1649 ;       Registers R1, R3, R4, and R5 are preserved.
                        0486  1650 ;
                        0486  1651 ;--
                        0486  1652
                        0486  1653 EXE$ALLOC_SPTS::
                  3A    BB    0486  1654          PUSHR   #^M<R1,R3,R4,R5>        ; Save registers.
     50    2044 8F    3C    0488  1655          MOVZWL  #SS$_INSFSPTS,R0        ; Assume allocation failure.
           53   62    D0    048D  1656          MOVL    CIN$L_SPTCOUNT(R2),R3   ; Get number of SPTs needed.
     51    00000000'GF    D0    0490  1657          MOVL    G^EXE$GL_RTBITMAP,R1    ; Get address of bit map
```

```
                             0497                                         ; control block.
                   60    13  0497          BEQL    60$                      ; If none, no SPTs available.
          04 A1    53    D1  0499          CMPL    R3,RBM$L_FREECOUNT(R1)   ; Are there enough SPTs left?
                   5A    14  049D          BGTR    60$                      ; No. Return with failure.
                   54    D4  049F          CLRL    R4                       ; Clear starting bit position.
                             04A1
                             04A1  10$:
        55    54   53    C1  04A1          ADDL3   R3,R4,R5                 ; Calculate highest bit
                             04A5                                           ; position needed in scan.
    00000000'GF    55    D1  04A5          CMPL    R5,G^EXE$GL_RTIMESPT     ; Is it higher than allowed?
                   4B    14  04AC          BGTR    60$                      ; Yes. Return with failure.
             20    54    EA  04AE          FFS     R4,#32,-                 ; Look for a free SPT (a set
          54    0C A1        04B1                  RBM$L_BITMAP(R1),R4      ; bit).
                   EB    13  04B4          BEQL    10$                      ; If none, go to next longword.
        55    54   53    C1  04B6          ADDL3   R3,R4,R5                 ; Again, calculate highest bit
                             04BA                                           ; position needed in scan.
          04 A2    54    D0  04BA          MOVL    R4,CIN$L_STARTBIT(R2)    ; Save starting bit number.
                             04BE
                             04BE  20$:
             20    54    EB  04BE          FFC     R4,#32,-                 ; Find first allocated SPT (a
          54    0C A1        04C1                  RBM$L_BITMAP(R1),R4      ; clear bit).
             55    54    D1  04C4          CMPL    R4,R5                    ; Past the highest bit needed?
                   07    18  04C7          BGEQ    30$                      ; Yes. Branch with success.
       F0 0C A1    54    E0  04C9          BBS     R4,RBM$L_BITMAP(R1),20$  ; If no clear bit found yet,
                             04CE                                           ; continue this scan.
                   D1    11  04CE          BRB     10$                      ; Otherwise, restart scan.
                             04D0
                             04D0  30$:
        50    04 A2      D0  04D0          MOVL    CIN$L_STARTBIT(R2),R0    ; Get starting bit number.
             61    50    C1  04D4          ADDL3   R0,RBM$L_STARTVPN(R1),-  ; Calculate the VPN of the
             04 A2           04D7                  CIN$L_STARTVPN(R2)       ; first SPT allocated.
                             04D9
                             04D9  ;
                             04D9  ; Allocate the SPTs by clearing the appropriate bits in the SPT bit
                             04D9  ; map.
                             04D9  ;
                             04D9  ; Registers are as follows:
                             04D9  ;
                             04D9  ;       R0      - starting bit number
                             04D9  ;       R1      - address of the real time bit map
                             04D9  ;       R2      - address of the quadword descriptor
                             04D9  ;       R3      - number of bits to alter
                             04D9  ;
                             04D9
                             04D9  40$:
             53    20    D1  04D9          CMPL    #32,R3                   ; Get number of bits to alter.
                   0E    18  04DC          BGEQ    50$                      ; Branch if 32 or less.
        20    50   00    F0  04DE          INSV    #0,R0,#32,-              ; Allocate the bits (by
             0C A1           04E2                  RBM$L_BITMAP(R1)         ; clearing them).
             50    20    C0  04E4          ADDL    #32,R0                   ; Move to next longword.
             53    20    C2  04E7          SUBL    #32,R3                   ; Subtract out number of bits
                             04EA                                           ; altered.
                   ED    11  04EA          BRB     40$                      ; Go alter more bits.
                             04EC
                             04EC  50$:
  0C A1  53    50  00    F0  04EC          INSV    #0,R0,R3,-               ; Allocate the bits (by
                             04F2                  RBM$L_BITMAP(R1)         ; clearing them.
```

```
                     04F2  1715
                     04F2  1716 ;
                     04F2  1717 ; Return with success.
                     04F2  1718 ;
                     04F2  1719
      62     C2      04F2  1720          SUBL    CIN$L_SPTCOUNT(R2),-    ; Reduce free count by number
   04 A1             04F4  1721                  RBM$L_FREECOUNT(R1)     ; allocated.
50    01     3C      04F6  1722          MOVZWL  #SS$_NORMAL,R0          ; Set success status code.
                     04F9  1723
                     04F9  1724 60$:
      3A     BA      04F9  1725          POPR    #^M<R1,R3,R4,R5>        ; Restore registers.
             05      04FB  1726          RSB                             ; Return.
```

```
                                                J 1
                     04FC  1728          .SBTTL  EXE$SETUP_SPTS, Validate and set access rights to SPTs
                     04FC  1729
                     04FC  1730 ;++
                     04FC  1731 ; EXE$SETUP_SPTS - Initialize SPTs to double map user's buffer
                     04FC  1732 ;
                     04FC  1733 ; Functional description:
                     04FC  1734 ;
                     04FC  1735 ;       When this routine gains control, IPL is at driver fork level.
                     04FC  1736 ;
                     04FC  1737 ;       This routine sets the valid bits and requested access bits in
                     04FC  1738 ;       a contiguous set of SPTs.
                     04FC  1739 ;
                     04FC  1740 ; Inputs:
                     04FC  1741 ;
                     04FC  1742 ;       R0      - access mask for pages
                     04FC  1743 ;       R1      - process address of the user's buffer
                     04FC  1744 ;       R2      - address of quadword descriptor of SPTs:
                     04FC  1745 ;
                     04FC  1746 ;                       CIN$L_SPTCOUNT(R2) - number of SPTs to validate
                     04FC  1747 ;                       CIN$L_STARTVPN(R2) - starting VPN
                     04FC  1748 ;
                     04FC  1749 ; Outputs:
                     04FC  1750 ;
                     04FC  1751 ;       The routine preserves all registers.
                     04FC  1752 ;
                     04FC  1753 ;--
                     04FC  1754
                     04FC  1755 EXE$SETUP_SPTS::
          007F 8F    BB  04FC  1756          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6>; Save some registers.
    54      04 A2    D0  0500  1757          MOVL    CIN$L_STARTVPN(R2),R4   ; Get starting VPN.
       56    62    D0  0504  1758          MOVL    CIN$L_SPTCOUNT(R2),R6   ; Get number of SPTs to setup.
       52    51    D0  0507  1759          MOVL    R1,R2                   ; Move process address.
                     050A  1760
                     050A  1761 ;
                     050A  1762 ; Calculate the address of the system page table entry that corresponds
                     050A  1763 ; to the starting VPN of the system-mapped buffer.
                     050A  1764 ;
                     050A  1765
    53    00000000'GF    D0  050A  1766          MOVL    G^MMG$GL_SPTBASE,R3     ; Get base of system page table.
       51    6344    DE  0511  1767          MOVAL   (R3)[R4],R1             ; Get address of SPT for VPN.
                     0515  1768
                     0515  1769 ;
                     0515  1770 ; Obtain the process page table entry of the next page in the user's
                     0515  1771 ; buffer.
                     0515  1772 ;
                     0515  1773
    54      00B0 C5    D0  0515  1774          MOVL    UCB$L_CI_PCB(R5),R4     ; Get process PCB address.
       55    6C A4    D0  051A  1775          MOVL    PCB$L_PHD(R4),R5        ; Get process PHD address.
                     051E  1776
                     051E  1777 10$:
    00000000'GF    16  051E  1778          JSB     G^MMG$PTEADRCHK         ; Get process PTE for this page.
                     0524  1779
                     0524  1780 ;
                     0524  1781 ; Register usage is now the following:
                     0524  1782 ;
                     0524  1783 ;       R0      - status from MMG$PTEADRCHK
                     0524  1784 ;       R1      - preserved; address of SPT for current VPN
```

```
                                                K 1
                     0524  1785 ;       R2      - preserved; process virtual address
                     0524  1786 ;       R3      - system virtual address of process page table entry
                     0524  1787 ;       R4      - preserved; address of the PCB (process control block)
                     0524  1788 ;       R5      - preserved; address of the PHD (process header block)
                     0524  1789 ;       R6      - preserved; count of SPTs left to setup
                     0524  1790 ;
                     0524  1791 ;       (SP)    - preserved; mask of page validation for the page
                     0524  1792 ;
                     0524  1793
           16 50  E9 0524  1794         BLBC    R0,20$                  ; Branch to exit on error.
                     0527  1795
                     0527  1796 ;
                     0527  1797 ; Get the physical page frame number from the process page table entry
                     0527  1798 ; for the page. Insert this and the validation mask in the SPT.
                     0527  1799 ;
                     0527  1800
        00   EF      0527  1801         EXTZV   #PTE$V_PFN,-            ; Extract the page frame number
  53 63  15          0529  1802                 #PTE$S_PFN,(R3),R3      ; of this page.
  81 53  6E  C9      052C  1803         BISL3   (SP),R3,(R1)+           ; Set up page table entry.
                     0530  1804
                     0530  1805 ;
                     0530  1806 ; See if more SPTs to setup. If not, invalidate the translation buffer,
                     0530  1807 ; and return to caller with success status.
                     0530  1808 ;
                     0530  1809
52 00000200 8F   C0  0530  1810         ADDL    #^X200,R2               ; Increment process address by
                     0537  1811                                         ; one page.
        E4 56    F5  0537  1812         SOBGTR  R6,10$                  ; Loop if more to do.
                     053A  1813         INVALID                         ; Clear translation buffer.
                     053A                       .IF B
39 00    DA          053A                       MTPR    #0,S^#PR$_TBIA
                     053D                       .IFF
                     053D                       .IF B
                     053D                       MTPR    ,S^#PR$_TBIS
                     053D                       .IFF
                     053D                       MOVL    ,
                     053D                       MTPR    ,S^#PR$_TBIS
                     053D                       .ENDC
                     053D                       .ENDC
                     053D
                     053D  1814
                     053D  1815 20$:
007F 8F  BA          053D  1816         POPR    #^M<R0,R1,R2,R3,R4,R5,R6>
         05          0541  1817         RSB                             ; Restore registers and return.
```

```
                                                L 1
                     0542  1819          .SBTTL  EXE$DEAL_SPTS, Deallocate real time SPTs
                     0542  1820
                     0542  1821 ;++
                     0542  1822 ; EXE$DEAL_SPTS - Deallocate SPTs used to double map process buffer
                     0542  1823 ;
                     0542  1824 ; Functional description:
                     0542  1825 ;
                     0542  1826 ;       When this routine gains control, IPL is at driver fork level.
                     0542  1827 ;
                     0542  1828 ;       Using a bit map whose address is stored in the control block
                     0542  1829 ;       addressed by EXE$GL_RTBITMAP, deallocate "n" contiguous SPTs.
                     0542  1830 ;
                     0542  1831 ; Inputs:
                     0542  1832 ;
                     0542  1833 ;       R2      - address of a quadword descriptor:
                     0542  1834 ;
                     0542  1835 ;                       CIN$L_SPTCOUNT(R2) - number of SPTs allocated
                     0542  1836 ;                       CIN$L_STARTVPN(R2) - starting VPN
                     0542  1837 ;
                     0542  1838 ; Implicit inputs:
                     0542  1839 ;
                     0542  1840 ;       EXE$GL_RTBITMAP - address of SPT bit map control block.
                     0542  1841 ;
                     0542  1842 ;       In the bit map, unset bits are allocated SPTs.
                     0542  1843 ;
                     0542  1844 ; Outputs:
                     0542  1845 ;
                     0542  1846 ;       The routine preserves all registers except R0.
                     0542  1847 ;
                     0542  1848 ;--
                     0542  1849
                     0542  1850 EXE$DEAL_SPTS::
                 0B  BB  0542  1851          PUSHR   #^M<R0,R1,R3>           ; Save registers.
      51 00000000'GF  D0  0544  1852          MOVL    G^EXE$GL_RTBITMAP,R1    ; Get address of bit map
                     0548  1853                                              ; control block.
                 61  C3  054B  1854          SUBL3   RBM$L_STARTVPN(R1),-    ; Calculate the starting bit
               50  04  A2  054D  1855                  CIN$L_STARTVPN(R2),R0  ; number of the allocated bits.
                 53  62  D0  0550  1856          MOVL    CIN$L_SPTCOUNT(R2),R3 ; Get number of bits.
                     0553  1857
                     0553  1858 10$:
                 53  20  D1  0553  1859          CMPL    #32,R3                  ; Branch if number of bits left
                     12  18  0556  1860          BGEQ    20$                     ; to alter is 32 or less.
      20 50 FFFFFFFF  8F  F0  0558  1861          INSV    #-1,R0,#32,-            ; Deallocate the bits by 32.
                   0C  A1  0560  1862                  RBM$L_BITMAP(R1)
                 50  20  C0  0562  1863          ADDL    #32,R0                  ; Move to next longword.
                 53  20  C2  0565  1864          SUBL    #32,R3                  ; Subtract out number of bits
                     0568  1865                                                  ; altered.
                     E9  11  0568  1866          BRB     10$                     ; Try for more.
                     056A  1867
                     056A  1868 20$:
0C A1 53 50 FFFFFFFF  8F  F0  056A  1869          INSV    #-1,R0,R3,-             ; Deallocate the remaining bits.
                     0574  1870                  RBM$L_BITMAP(R1)
                 62  C0  0574  1871          ADDL    CIN$L_SPTCOUNT(R2),-    ; Recalculate number of free
               04  A1  0576  1872                  RBM$L_FREECOUNT(R1)     ; SPTs.
                 0B  BA  0578  1873          POPR    #^M<R0,R1,R3>           ; RESTORE REGISTERS
                     05  057A  1874          RSB                             ; Return to caller.
```

```
                                                M 1
057B  1876          .SBTTL  CI_END, End of driver
057B  1877
057B  1878 ;++
057B  1879 ; Label that marks the end of the driver
057B  1880 ;--
057B  1881
057B  1882 CI_END:                                      ; Last location in driver
057B  1883          .END
```

Symbol table

```
$$$                     = 00000020 R     02
$$OP                    = 00000C02
ACB$B_RMOD              = 0000000B
ACB$B_TYPE              = 0000000A
ACB$K_LENGTH            = 0000001C
ACB$L_AST               = 00000010
ACB$L_ASTPRM            = 00000014
ACB$L_ASTQFL            = 00000000
ACB$L_PID               = 0000000C
ACB$M_QUOTA             = 00000040
ACB$W_SIZE              = 00000008
AST_COUNT               = 00000014
AST_PARAMETER           = 00000010
AST_ROUTINE             = 0000000C
AT$_UBA                 = 00000001
BUFFER_DESC             = 00000000
CANCEL_EXIT               00000421 R     03
CHECK_AST                 0000032E R     03
CI$DDT                    00000000 RG    03
CIN$L_CANCEL            = 0000000C
CIN$L_INIDEV            = 00000000
CIN$L_ISR               = 00000008
CIN$L_SPTCOUNT          = 00000000
CIN$L_START             = 00000004
CIN$L_STARTBIT          = 00000004
CIN$L_STARTVPN          = 00000004
CIN$M_AST               = 00000008
CIN$M_CANCEL            = 00000080
CIN$M_EFN               = 00000001
CIN$M_INIDEV            = 00000010
CIN$M_ISR               = 00000040
CIN$M_REPEAT            = 00000004
CIN$M_START             = 00000020
CIN$M_USECAL            = 00000002
CIN$S_EFNUM             = 00000010
CIN$V_CANCEL            = 00000007
CIN$V_EFN               = 00000000
CIN$V_EFNUM             = 00000010
CIN$V_INIDEV            = 00000004
CIN$V_ISR               = 00000006
CIN$V_START             = 00000005
CIN$V_USECAL            = 00000001
CI_ALLOC_ASTS             000002C0 R     03
CI_CANCEL                 000003F2 R     03
CI_CONNECT                0000007F R     03
CI_DISCONNECT             00000422 R     03
CI_DUMMY_RSB              00000482 R     03
CI_END                    0000057B R     03
CI_FORCE_CANCEL           000003FD R     03
CI_FORK_PROCESS           0000034F R     03
CI_FUNCTABLE              00000038 R     03
CI_INIT_DEVICE            00000054 R     03
CI_INTERRUPT              00000312 R     03
CI_ISR_CALL               00000328 R     03
CI_START                  00000302 R     03
CI_START_CALL             0000030C R     03
CRB$L_INTD              = 00000024
DDB$L_DDT               = 0000000C
DDB$L_UCB               = 00000004
DEV$M_AVL               = 00040000
DEV$M_RTM               = 20000000
DISMISS_INT               00000325 R     03
DOUBLE_MAP                00000120 R     03
DPT$C_LENGTH            = 00000038
DPT$C_VERSION           = 00000004
DPT$INITAB                00000038 R     02
DPT$REINITAB              0000004E R     02
DPT$TAB                   00000000 R     02
DYN$C_ACB               = 00000002
DYN$C_CRB               = 00000005
DYN$C_DDB               = 00000006
DYN$C_DPT               = 0000001E
DYN$C_UCB               = 00000010
ENTRY_LIST              = 00000004
ERROR                     000002B5 R     03
ERROR_DEALSPTS            0000029C R     03
EXE$ABORTIO               ********   X   03
EXE$ALLOC_SPTS            00000486 RG    03
EXE$ALONONPAGED           ********   X   03
EXE$DEAL_SPTS             00000542 RG    03
EXE$DEANONPAGED           ********   X   03
EXE$FORK                  ********   X   03
EXE$GL_RTBITMAP           ********   X   03
EXE$GL_RTIMESPT           ********   X   03
EXE$MODIFYLOCK            ********   X   03
EXE$QIODRVPKT             ********   X   03
EXE$SETUP_SPTS            000004FC RG    03
EXE$WRITELOCK             ********   X   03
FLAGS                   = 00000008
FUNCTAB_LEN             = 0000001C
IDB$L_CSR               = 00000000
IDB$L_OWNER             = 00000004
IO$_CONINTREAD          = 0000003C
IO$_CONINTWRITE         = 0000003D
IO$_VIRTUAL             = 0000003F
IOC$CANCELIO              ********   X   03
IOC$MNTVER                ********   X   03
IOC$REQCOM                ********   X   03
IOC$RETURN                ********   X   03
IO_COMPLETE               000003E7 R     03
IRP$S_FCODE             = 00000006
IRP$V_FCODE             = 00000000
IRP$W_FUNC              = 00000020
LOCK_PAGES                00000100 R     03
LOOP                      000002C7 R     03
MASKH                   = 30000000
MASKL                   = 00000000
MMG$GL_SPTBASE            ********   X   03
MMG$PTEADRCHK             ********   X   03
P1                      = 00000000
P2                      = 00000004
P3                      = 00000008
P4                      = 0000000C
P5                      = 00000010
```

```
P6                              =  00000014
PCB$L_PHD                       =  0000006C
PCB$L_PID                       =  00000060
PCB$Q_PRIV                      =  00000084
PCB$W_ASTCNT                    =  00000038
PR$_IPL                         =  00000012
PR$_TBIA                        =  00000039
PRI$_IOCOM                      =  00000001
PRV$V_CMKRNL                    =  00000000
PSL$S_PRVMOD                    =  00000002
PSL$V_PRVMOD                    =  00000016
PTE$C_KR                        =  18000000
PTE$C_KW                        =  10000000
PTE$M_VALID                     =  80000000
PTE$S_PFN                       =  00000015
PTE$V_PFN                       =  00000000
QUEUE_PACKET                       00000291 R      03
RBM$L_BITMAP                    =  0000000C
RBM$L_FREECOUNT                 =  00000004
RBM$L_STARTVPN                  =  00000000
SCH$CREF                           ********   X    03
SCH$POSTEF                         ********   X    03
SCH$QAST                           ********   X    03
SETUP_ASTS                         0000022A R      03
SETUP_ENTRIES                      00000187 R      03
SIZ...                          =  00000001
SS$_ACCVIO                      =  0000000C
SS$_BADPARAM                    =  00000014
SS$_DISCONNECT                  =  0000204C
SS$_EXQUOTA                     =  0000001C
SS$_INSFSPTS                    =  00002044
SS$_NOPRIV                      =  00000024
SS$_NORMAL                      =  00000001
UCB$B_CI_ASTMOD                    00000098
UCB$B_CI_SPARE                     00000099
UCB$B_DIPL                      =  0000005E
UCB$B_FIPL                      =  0000000B
UCB$K_CI_ISARGC                 =  00000005
UCB$K_CI_LENGTH                    00000100
UCB$K_CI_STARGC                 =  00000004
UCB$K_LENGTH                    =  00000090
UCB$L_CI_ABLINK                    000000AC
UCB$L_CI_AFLINK                    000000A8
UCB$L_CI_AST                       0000009C
UCB$L_CI_ASTPRM                    000000A0
UCB$L_CI_CANCEL                    000000D0
UCB$L_CI_INIDEV                    000000BC
UCB$L_CI_ISARG1                    000000EC
UCB$L_CI_ISARG2                    000000F0
UCB$L_CI_ISARG3                    000000F4
UCB$L_CI_ISARG4                    000000F8
UCB$L_CI_ISARG5                    000000FC
UCB$L_CI_ISARGC                    000000E8
UCB$L_CI_ISR                       000000C8
UCB$L_CI_ISRCAL                    000000CC
UCB$L_CI_PCB                       000000B0
UCB$L_CI_STACAL                    000000C4
UCB$L_CI_STARG1                    000000D8
UCB$L_CI_STARG2                    000000DC
UCB$L_CI_STARG3                    000000E0
UCB$L_CI_STARG4                    000000E4
UCB$L_CI_STARGC                    000000D4
UCB$L_CI_START                     000000C0
UCB$L_CRB                       =  00000024
UCB$L_DEVCHAR                   =  00000038
UCB$L_DEVDEPEND                 =  00000044
UCB$L_IRP                       =  00000058
UCB$M_CI_AST                    =  00000008
UCB$M_CI_CANCEL                 =  00000080
UCB$M_CI_EFN                    =  00000001
UCB$M_CI_INIDEV                 =  00000010
UCB$M_CI_ISR                    =  00000040
UCB$M_CI_REPEAT                 =  00000004
UCB$M_CI_START                  =  00000020
UCB$M_CI_USECAL                 =  00000002
UCB$M_ONLINE                    =  00000010
UCB$Q_CI_BUFDSC                    00000090
UCB$Q_CI_SPTDSC                    000000B4
UCB$V_BSY                       =  00000008
UCB$V_CANCEL                    =  00000003
UCB$V_CI_AST                    =  00000003
UCB$V_CI_CANCEL                 =  00000007
UCB$V_CI_EFN                    =  00000000
UCB$V_CI_INIDEV                 =  00000004
UCB$V_CI_REPEAT                 =  00000002
UCB$V_CI_UCBFRK                 =  00000008
UCB$V_CI_USECAL                 =  00000001
UCB$W_CI_ACBCNT                    000000A4
UCB$W_CI_ACBNOW                    000000A6
UCB$W_CI_EFNUM                     0000009A
UCB$W_STS                       =  00000064
VA$M_SYSTEM                     =  80000000
VA$S_BYTE                       =  00000009
VA$V_BYTE                       =  00000000
VEC$L_IDB                       =  00000008
VEC$L_INITIAL                   =  0000000C
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000100 (256.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	00000072 (114.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	0000057B (1403.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.06	00:00:01.33
Command processing	146	00:00:00.45	00:00:04.31
Pass 1	587	00:00:17.08	00:01:11.64
Symbol table sort	0	00:00:02.52	00:00:11.75
Pass 2	328	00:00:04.33	00:00:23.77
Symbol table output	24	00:00:00.14	00:00:00.41
Psect synopsis output	3	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1119	00:00:24.59	00:01:53.23

The working set limit was 2400 pages.
145741 bytes (285 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2393 non-local and 49 local symbols.
1883 source lines were read in Pass 1, producing 18 object records in Pass 2.
45 pages of virtual memory were used to define 42 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	28
_255DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	40

2652 GETS were required to define 40 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:CONINTERR/OBJ=OBJ$:CONINTERR MSRC$:CONINTERR/UPDATE=(ENH$:CONINTERR)+EXECML$/LIB
```

XADRIVER
MAR

CONINTERR
LIS

XIDRIVER
MAR

PAMAC
MAR

CNDRIVER
LIS

DBDRIVER
LIS

CRDRIVER
LIS

CVDRIVER
LIS

DDDRIVER
LIS

DDDMP
LIS